Reinforcement Learning: Tutorial

Clemens Heitzinger

June 6, 2025

Contents

1 Tutorial on 27 March 2025 (18 Exercises)

1. **ENV/EX** – *Think of application.*
   Think of a (preferably creative) application of reinforcement learning. Specify the states, actions, and rewards as well as what is needed to satisfy the Markov property.

2. **ENV/COUNTEREX** – *Goal-directed learning task that is not an MDP.*
   Try to find a goal-directed learning task that cannot be represented by a Markov decision process.

3. **AS** – *ϵ-greedy action selection.*
   Assume that ϵ-greedy action selection is used.

   (a) Suppose $|\mathcal{A}| = 4$ and $\epsilon = 0.2$. When using ϵ-greedy action selection, what is the probability that the greedy action is selected?

   (b) Which value of ϵ would achieve a probability of 70% of selecting the greedy action?

   (c) Generalize the formula for calculating the probability of selecting the greedy action in ϵ-greedy action selection for any $|\mathcal{A}|$ and any ϵ.

4. **STS/H** – *Harmonic step sizes.*
   Show that the step sizes

   $$\alpha_n := \frac{1}{an+b}, \qquad a, b \in \mathbb{R},$$

   (where $a \in \mathbb{R}^+$ and $b \in \mathbb{R}$ are chosen such that $an + b \neq 0$) satisfy the convergence conditions

   $$\sum_{n=1}^{\infty} \alpha_n = \infty, \qquad \sum_{n=1}^{\infty} \alpha_n^2 < \infty.$$

5. **STS/U** – *Unbiased step sizes.*
   We use the iteration

   $$\begin{aligned} &Q_1 \in \mathbb{R}, \\ &Q_{n+1} := Q_n + \alpha_n (R_n - Q_n), \qquad n \geq 1, \end{aligned}$$

   to estimate Q_n using R_n, where

   $$\alpha_n := \frac{\alpha}{\beta_n}, \qquad \alpha \in (0,1), \quad n \geq 1,$$

   and

   $$\begin{aligned} &\beta_0 := 0, \\ &\beta_n := \beta_{n-1} + \alpha(1 - \beta_{n-1}), \qquad n \geq 1. \end{aligned}$$

Show that the iteration for Q_n above yields an exponential recency-weighted average *without initial bias* (i.e., the Q_n do not depend on the initial value Q_1).

6. **MAB/EPS** – *Multi-armed bandits with ϵ-greedy action selection (programming).*
   You play against a 10-armed bandit, where at the beginning of each episode the true value $q_*(a)$, $a \in \{1, \dots, 10\}$, of each of the 10 actions is chosen to be normally distributed with mean zero and unit variance. The rewards after choosing action/bandit a are normally distributed with mean $q_*(a)$ and unit variance. Using the simple bandit algorithm and ϵ-greedy action selection, you have 1000 time steps or tries in each episode to maximize the average reward starting from zero knowledge about the bandits.

   Which value of ϵ maximizes the average reward? Which value of ϵ maximizes the percentage of optimal actions taken?

7. **MAB/UCB** – *Multi-armed bandits with upper-confidence-bound action selection (programming).*
   This exercise is the same as in Exercise MAB/EPS, but now the actions

   $$A_t := \arg\max_a \left(Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right)$$

   are selected according to the upper-confidence bound.

   Which value of c yields the largest average reward?

8. **MAB/SOFTMAX** – *Multi-armed bandits with soft-max action selection (programming).*
   This exercise is the same as Exercise MAB/EPS, but now the actions $A_t \in \mathcal{A} = \{1, \dots, |\mathcal{A}|\}$ are selected with probability

   $$\mathbb{P}[a] = \frac{\exp(Q_t(a)/\tau)}{\sum_{i=1}^{|\mathcal{A}|} \exp(Q_t(i)/\tau)},$$

   where the parameter τ is called the temperature. This probability distribution is called the soft-max or Boltzmann distribution.

   What are the effects of low and high temperatures, i.e., how does the temperature influence the probability distribution all else being equal? Which value of τ yields the largest average reward?

9. **MDP/G1** – *Returns and episodes.*
   Suppose $\gamma := 1/2$ and the rewards $R_1 := 1$, $R_2 := -1$, $R_3 := 2$, $R_4 := -1$, and $R_5 := 2$ are received in an episode with length $T := 5$. What are $G_0, \dots, G_5$?

10. MDP/G2 – *Returns and episodes.*
    Suppose $\gamma := 0.9$ and the reward sequence starts with $R_1 := -1$ and $R_2 := 2$ and is followed by an infinite sequence of 1s. What are G_0, G_1, and G_2?

11. MDP/V – *Equation for v_π.*
    Give an equation for v_π in terms of q_π and π.

12. MDP/Q – *Equation for q_π.*
    Give an equation for q_π in terms of v_π and the four-argument p.

13. MDP/RET – *Change of return.*
    In episodic tasks and in continuing tasks, how does the return G_t change if a constant c is added to all rewards R_t?

14. MDP/BELLMAN/QPI – *Bellman equation for q_π.*
    Analogous to the derivation of the Bellman equation for v_π, derive the Bellman equation for q_π.

15. MDP/VSTAR – *Equation for v_*.*
    Give an equation for v_* in terms of q_*.

16. MDP/QSTAR – *Equation for q_*.*
    Give an equation for q_* in terms of v_* and the four-argument p.

17. MDP/PISTAR/VSTAR – *Equation for π_*.*
    Give an equation for π_* in terms of q_*.

18. MDP/PISTAR/QSTAR – *Equation for π_*.*
    Give an equation for π_* in terms of v_* and the four-argument p.

2 Tutorial on 3 & 10 April 2025 (7 Exercises)

1. **DP/BANACH** – *Formulate Banach fixed-point theorem.*
   Formulate the Banach fixed-point theorem after defining all relevant terms.

2. **DP/BANACH/PROOF** – *Prove Banach fixed-point theorem.*
   Prove the Banach fixed-point theorem.

3. **DP/UPDATE/Q** – *Update rule for q_π.*
   Using the Bellman equation for q_π (see Exercise MDP/BELLMAN/QPI), find an update rule for the approximation q_{k+1} of q_π (in terms of q_k, π, and p) analogous to the update rule for v_{k+1}.

4. **GW/SIMPLE** – *Simple 4×4 grid world (programming).*
   Implement a 4×4 grid world with two terminal states in the upper left corner and lower right corners (resulting in 14 non-terminal states). The four actions $\mathcal{A} = \{\text{up}, \text{down}, \text{left}, \text{right}\}$ act deterministically, the discount factor is $\gamma = 1$, and the reward is always equal to -1. Ensure that a maximum number of time steps can be specified.

5. **DP/POLICY/EVAL** – *Iterative policy evaluation (programming).*
   Implement iterative policy evaluation and use it to estimate v_π for the grid world in Exercise GW/SIMPLE, where π is the equiprobable random policy.

6. **DP/POLICY/ITER** – *Policy iteration (programming).*
   Implement policy iteration and use it to estimate π_* for the grid world in Exercise GW/SIMPLE.

7. **DP/VALUE/ITER** – *Value iteration (programming).*
   Implement value iteration and use it to estimate π_* for the grid world in Exercise GW/SIMPLE.

3 Tutorial on 8 May 2025 (8 Exercises)

1. **OP/WIS/REC** – *Recursive formula for weighted importance sampling for off-policy learning.*
   In weighted importance sampling, we calculate the estimate

   $$V^\pi_{n+1} := \frac{\sum_{k=1}^n W_k G_k}{\sum_{k=1}^n W_k}, \qquad n \ge 1,$$

   given the returns $G_1, G_2, \ldots$ and the weights $W_1, W_2, \ldots$.
   Show that the iteration

   $$V_{n+1} := V_n + \frac{W_n}{C_n}(G_n - V_n), \qquad n \ge 1,$$

   where

   $$\begin{aligned} C_0 &= 0, \\ C_{n+1} &:= C_n + W_{n+1}, \qquad n \ge 0, \end{aligned}$$

   yields the same values $V^\pi_n = V_n$ for all $n \in \{2, 3, \ldots\}$.
   Hint: Follow the derivation of the formula $Q_{n+1} = Q_n + (1/n)(R_n - Q_n)$.

2. **MC/CTRL/IMPL** – *Off-policy MC control (programming).*
   Implement off-policy MC control and use it to estimate π_* for the grid world in Exercise GW/SIMPLE.

3. **MC/CTRL/RATIO** – *Importance-sampling ratio in off-policy MC control.*
   In the off-policy MC-control algorithm in [1, Section 5.7], the importance-sampling ratio is updated according to

   $$W := \frac{W}{b(A_t|S_t)},$$

   although the definition of the importance-sampling ratio $\rho_{t:T-1}$ implies

   $$W := \frac{\pi(A_t|S_t)}{b(A_t|S_t)} W.$$

   Why is the update in the algorithm nevertheless correct?

4. **IMPL** – *Windy Grid World.*
   Implement Windy Grid World (see Section 2.4.2).

5. **IMPL** – *Cliff Walking.*
   Implement Cliff Walking (see Section 2.4.3).

6. **IMPL** – *Frozen Lake.*
   Implement Frozen Lake (see Section 2.4.4).

7. **IMPL** – *Blackjack.*
   Implement Blackjack (see Section 2.4.5).

8. **IMPL** – *Autoregressive Trend Process.*
   Implement an autoregressive trend process (see Section 2.4.7).

4 Tutorial on 5 June 2025 (9 Exercises)

1. OP/QLEARNING – *Q-learning is off-policy.*
   Why is Q-learning considered an *off-policy* control method?

2. TD/SARSA/QLEARNING – *Difference between SARSA and Q-learning.*
   Is Q-learning the same algorithm as SARSA with greedy action selection? If not, why?

3. TD/TAB/GRAD – *Tabular updates as gradient descent.*
   In order to (locally) minimize a function $f\colon \mathbb{R}^d \to \mathbb{R}$, we can use the gradient-descent iteration

   $$\mathbf{x}_{t+1} := \mathbf{x}_t - \alpha_t \nabla f(\mathbf{x}_t).$$

   Find a function f whose gradient-descent iteration is the general tabular update formula

   $$V(S_t) := V(S_t) + \alpha_t (Y_t - V(S_t)),$$

   while assuming that the gradient of the target Y_t vanishes.

4. GRAD/HUBER – *Huber loss and gradient descent.*
   Here, the 2-norm and the 1-norm are combined into a loss function. The Huber loss $L_\delta\colon \mathbb{R} \to \mathbb{R}_0^+$ is defined as

   $$L_\delta(a) := \begin{cases} \frac{1}{2}a^2, & |a| \le \delta, \\ \lambda|a| + \mu, & |a| > \delta, \end{cases}$$

   where $\delta \in \mathbb{R}_0^+$ is a constant that defines the transition from quadratic to linear behavior.

   (a) Determine the constants $\lambda, \mu \in \mathbb{R}$ such that L_δ is as smooth as possible.

   (b) For

   $$f(\mathbf{w}_t) := L_\delta(Y_t - \hat{v}(s, \mathbf{w}_t))$$

   in Exercise TDTABGRAD, what is the resulting tabular update formula?

   (c) For more than one sample, we can consider the loss $\sum_{i=1}^{d} L_\delta(a_i)$ of a vector $\mathbf{a} \in \mathbb{R}^d$. What are its advantages compared to the 2-norm and the 1-norm when used in gradient descent?

5. IMPL – *SARSA.*
   Implement SARSA and apply it to (at least) one of the environments you have implemented.

6. **IMPL** – *Expected SARSA.*
   Implement expected SARSA and apply it to (at least) one of the environments you have implemented.

7. **IMPL** – *Q-learning.*
   Implement Q-learning and apply it to (at least) one of the environments you have implemented.

8. **IMPL** – *Double Q-learning.*
   Implement double Q-learning and apply it to (at least) one of the environments you have implemented.

9. **TD/TAB/SPEEDYQLEARNING** – *Tabular Speedy Q-learning (programming).*
   In [2], the Q-learning variant

$$\begin{aligned}
Q_{t+1}(s,a) &:= Q_t(s,a) + \alpha_t(\mathcal{B}_t Q_{t-1}(s,a) - Q_t(s,a)) \\
&\quad + (1-\alpha_t)(\mathcal{B}_t Q_t(s,a) - \mathcal{B}_t Q_{t-1}(s,a)), \\
\alpha_t &:= \frac{1}{t+1}, \\
\mathcal{B}_t Q_t(s,a) &:= R_{t+1} + \gamma \max_a Q_t(S_{t+1},a),
\end{aligned}$$

   was defined and called Speedy Q-learning. (Here $\mathcal{B}_t$ is the (usual) empirical Bellman operator. Note that it is applied to $Q_t(s,a)$ and to $Q_{t-1}(s,a)$.)

   Implement Speedy Q-learning for estimating q_* and use it to estimate π_* for cliff walking in Exercise GW/CLIFF.

   Compare Q-learning and Speedy Q-learning with the learning rate defined above and with adjusted learning rates. Which algorithms converges faster? What is the influence of the learning rate?

5 Tutorial on 12 June 2025 (10 Exercises)

1. **TD/NSTEP/ERROR** – *n-step TD error as sum of TD errors.*
   Show that the n-step error $G_{t:t+n} - V_{t+n-1}(S_t)$ in the n-step TD update

   $$V_{t+n}(S_t) = V_{t+n-1}(S_t) + \alpha(G_{t:t+n} - V_{t+n-1}(S_t))$$

   can be written as a sum of TD errors

   $$\delta_t := R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

   assuming that the value estimates never change.

2. **TD/NSTEP/SARSA** – *n-step SARSA error as sum of TD errors.*
   Show that the n-step return of SARSA

   $$G_{t:t+n} := R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n})$$

   is equal to

   $$G_{t:t+n} = Q_{t-1}(S_t, A_t) + \sum_{k=t}^{\min(t+n,T)-1} \gamma^{k-t} \delta_k,$$

   where

   $$\delta_k := R_{k+1} + \gamma Q_k(S_{k+1}, A_{k+1}) - Q_{k-1}(S_k, A_k).$$

3. **APPROX/POLY** – *Orthogonal polynomials.*
   Give at least three kinds of (multivariate) orthogonal polynomials and discuss their respective advantages and disadvantages for approximating state-value and action-value functions.

4. **APPROX/HASHING** – *Tile coding using hashing.*
   Explain at least one hashing algorithm and how it can be applied to tile (coarse) coding.

5. **LSTD** – *Least-squares temporal difference (programming).*
   Implement least-squares temporal difference and apply it to one of the environments you have already implemented. What are your feature vectors? Use an iterative solver (from a linear-algebra software library) for the linear system $\hat{A}_t \mathbf{w}_t = \hat{\mathbf{b}}_t$. In the case of a control problem, explore what happens when you update the weight vector $\mathbf{w}_t$ periodically and not in every time step. Finally, compare the performance to at least one other control algorithm you have already implemented.

6. **APPROX/BAIRD** – *Baird's counterexample.*
   Explain Baird's counterexample in [1, Section 11.2] in detail.

7. **APPROX/TSITSIKLISVANROY** – *Tsitsiklis and Van Roy's counterexample.*
   Explain Tsitsiklis and Van Roy's counterexample in [1, Section 11.2] in detail.

8. **ERROR/EX112** – *S&B Example 11.2: A-split example, showing the naiveté of the naive residual-gradient algorithm.*
   Explain [1, Example 11.2] in detail.

9. **ERROR/EX113** – *S&B Example 11.3: A-presplit example, a counterexample for the mean squared Bellman error.*
   Explain [1, Example 11.3] in detail.

10. **ERROR/EX114** – *S&B Example 11.4: counterexample to the learnability of the Bellman error.*
    Explain [1, Example 11.4] in detail.

References

[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: an Introduction.* The MIT Press, 2nd edition edition, 2018.

[2] Mohammad Ghavamzadeh, Hilbert J. Kappen, Mohammad G. Azar, and Rémi Munos. Speedy Q-learning. In J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24 (NIPS 2011)*, pages 2411–2419. Curran Associates, Inc., 2011.