Reinforcement Learning — Exercises Lectures 1-5

Instructions

This is the first exercise booklet for Reinforcement Learning. It covers both ungraded exercises to practice at home or during the tutorial sessions as well as graded homework exercises. The graded homework assignments are clearly marked. Please note the following:

- Make sure you deliver answers in a clear and structured format. LaTeXhas our preference. Messy handwritten answers will not be graded.
- Pre-pend the name of your TA to the file name you hand in and remember to put your name and student ID on the submission;
- The deadline for this first assignment is *September 20th 2019* and will cover the material of lectures 1-5 (all questions marked 'homework' in this booklet).

Contents

Lecture 0: Prior knowledge self-test

0.1 Linear algebra and multivariable derivatives.

Consider the following matrices and vectors:

$$A = \begin{pmatrix} a_{11} & 0 \\ 0 & a_{22} \end{pmatrix} \quad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \quad c = \begin{pmatrix} y - x^2 \\ \frac{\ln x}{y} \end{pmatrix} \quad d = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} \quad e = \begin{pmatrix} x \\ y \end{pmatrix}$$

1. Compute AB, AB^T, and $d^T Bd$.
2. Find the inverses of A and B.
3. Compute $\frac{\partial c}{\partial x}$ and $\frac{\partial c}{\partial e}$.
4. Consider the function $f(\boldsymbol{x}) = \sum_i^N ix_i$, which maps an N-dimensional vector of real numbers $\boldsymbol{x}$ to a real number. Find an expression for $\frac{\partial f}{\partial \boldsymbol{x}}$ in terms of integers 1 to N.

Solution

1. Just straightforward computation.

$$AB = \begin{pmatrix} a_{11}b_{11} & a_{11}b_{12} \\ a_{22}b_{21} & a_{22}b_{22} \end{pmatrix} \quad AB^T = \begin{pmatrix} a_{11}b_{11} & a_{11}b_{21} \\ a_{22}b_{12} & a_{22}b_{22} \end{pmatrix}$$

$$d^T Bd = d_1 b_{11} d_1 + d_1 b_{12} d_2 + d_2 b_{21} d_1 + d_2 b_{22} d_2$$

2. The inverse of a diagonal matrix is a diagonal matrix whose entries are the reciprocal of the original diagonal. The inverse of a more general 2x2 matrix $M$is given as $\frac{1}{\det M} M*$, where det(.) is the determinant and $M*$ the adjugate matrix of M.

$$A^{-1} = \begin{pmatrix} 1/a_{11} & 0 \\ 0 & 1/a_{22} \end{pmatrix} \quad B^{-1} = \frac{1}{b_{11}b_{22} - b_{12}b_{21}} \begin{pmatrix} b_{22} & -b_{12} \\ -b_{21} & b_{11} \end{pmatrix}$$

3. Matrix / vector derivatives are defined similarly to scalar derivatives, although we have to choose a layout. We choose the numerator layout (also called Jacobian formulation), in which we treat the matrix / vector in the numerator as being transposed. I.e. if v is an n-vector and w is an m-vector, then $\frac{\partial v}{\partial w}$ will be an $(n \times m)$ matrix.

$$\frac{\partial c}{\partial x} = \begin{pmatrix} -2x \\ \frac{1}{xy} \end{pmatrix} \qquad \frac{\partial c}{\partial e} = \begin{pmatrix} \frac{\partial(y-x^2)}{\partial x} & \frac{\partial(y-x^2)}{\partial y} \\ \frac{\partial(\frac{\ln x}{y})}{\partial x} & \frac{\partial(\frac{\ln x}{y})}{\partial y} \end{pmatrix} = \begin{pmatrix} -2x & 1 \\ \frac{1}{xy} & -\frac{\ln x}{y^2} \end{pmatrix}$$

4. As above, we treat $\boldsymbol{x}$ as if transposed (i.e. a row vector). Thus our derivative will have the following form:

$$\frac{\partial f}{\partial \boldsymbol{x}} = \frac{\partial f}{\partial(x_1, x_2, ..., x_N)} = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_N} \right)$$

Now we can evaluate a single such term either by inspection of the function $f(\boldsymbol{x}) = \sum_i^N ix_i$, or by explicit algebraic manipulation:

$$\frac{\partial f}{\partial x_j} = \frac{\partial \left(\sum_i^N ix_i\right)}{\partial x_j} = \sum_i^N \frac{\partial (ix_i)}{\partial x_j} = \sum_i^N i\delta_{ij} = j$$

where δ_{ij} is the Kronecker delta, which equals one if $i = j$ and zero otherwise. Plugging the result into our expression for $\frac{\partial f}{\partial \boldsymbol{x}}$ we obtain: $\frac{\partial f}{\partial \boldsymbol{x}} = (1, 2, ..., N)$ (i.e. the row-vector of integers 1 to N).

0.2 Probability theory

Assume X and Y are two independent random variables, respectively with mean μ, ν, and variance σ^2, τ^2.

1. What is the expected value of $X + \alpha Y$, where α is some constant?
2. What is the variance of $X + \alpha Y$?

Suppose now that we have a training set consisting of points $(x_1, x_2, ..., x_n)$ and real values y_i associated to each point x_i. Assume that there is a function f such that $y = f(x) + \epsilon$, where ϵ is a noise vector with zero mean and variance σ^2.

Assume we are looking for a function $\hat{f}$ that minimises the squared error $(y - \hat{f}(x))^2$. Recall the bias-variance decomposition of this squared error on an unseen sample x:

$$\mathrm{E}\left[(y - \hat{f}(x))^2\right] = \left(\mathrm{Bias}[\hat{f}(x)]\right)^2 + \mathrm{Var}[\hat{f}(x)] + \sigma^2 \tag{1}$$

where

$$\mathrm{Bias}[\hat{f}(x)] = \mathrm{E}[\hat{f}(x)] - f(x) \qquad \mathrm{Var}[\hat{f}(x)] = \mathrm{E}[\hat{f}(x)^2] - \mathrm{E}[\hat{f}(x)]^2$$

and the expectation $\mathrm{E}[.]$ ranges over different choices of the training set $(x_1, x_2, ..., x_n; y_1, y_2, ..., y_n)$, all sampled from the same joint distribution $P(X, Y)$.

3. Explain what errors the various terms in equation (1) represent (for instance, under what circumstances is a particular term large or small?).
4. Explain why this decomposition is also known as the 'bias-variance trade-off'.

Solution

1. From linearily of expectation we have $\mathrm{E}[X + aY] = \mathrm{E}[X] + a\mathrm{E}[Y] = \mu + a\nu$.
2. We know that for independent variables $\mathrm{Var}[aX + bY] = a^2\mathrm{Var}[X] + b^2\mathrm{Var}[Y]$. So $\mathrm{Var}[X + \alpha Y] = \sigma^2 + \alpha^2\tau^2$.

3. The bias term represents the error caused by the simplifying assumptions in our model. It will be large when we choose a model that is too simple to represent our training data, and small when our model is complex enough to fit the training data well. The variance term represents the sensitivity of our model to different sets of training data. It will be high when when our model is complex enough to fit many potential sets of training data well (probably fitting some of the noise), and small when our model is too simple. The final term represents the irreducible error. Whether it is high or low depends on how precisely our dataset was collected.

4. Typically, changes to our model that reduce bias will increase variance, and vice-versa. A model with small bias is strong enough to fit the training data, but this generally means it will also find some spurious correlations resulting from the precise training dataset drawn from $P(X, Y)$, which results in high variance. Similarly, a model with low variance does not fit these spurious correlations, but then generally will be worse at fitting different instances of the training data.

0.3 OLS, linear projection, and gradient descent.

Suppose now that we have a training set $\boldsymbol{X}$ consisting of n vectors (datapoints) of size m (features). Associated to this we have an n-dimensional vector of real values $\boldsymbol{y}$. Suppose we want to regress a linear model to this data using ordinary least-squares (OLS). That is, we fit linear model $f_\beta(\boldsymbol{X}) = \boldsymbol{X}\beta$, such that we minimise $(\boldsymbol{y} - f_\beta(\boldsymbol{X}))^2$ over the parameters β.

1. What is the dimensionality of the parameter vector β?
2. Show by differentiation that the OLS estimator $\hat{\beta}$ equals $(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$.

Another method for deriving the OLS estimator is by thinking geometrically. Imagine $\boldsymbol{y}$ and $\boldsymbol{X}\beta$ as vectors in an n-dimensional vector space. Further note that the regressors $\boldsymbol{X}\beta$ actually span an m-dimensional subspace of this larger vector space (the column space of $\boldsymbol{X}$), see Figure 1.

3. Define as $\boldsymbol{\epsilon}_\beta$ the residual vector $(\boldsymbol{y} - \boldsymbol{X}\beta)$. Argue that minimising the L_2 norm of this vector over β (as we do in OLS) is equivalent to choosing β such that $\boldsymbol{X}\beta = P(\boldsymbol{y})$, where $P(.)$ orthogonally projects $\boldsymbol{y}$ onto the linear subspace spanned by the regressors.
4. Argue that this is equivalent to setting $\boldsymbol{X}^T\boldsymbol{\epsilon}_\beta = 0$.
5. Show that from this also follows that $\hat{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$.

When our models get more complicated, we quickly lose the ability to optimise them analytically. Instead, we use numerical optimisation methods such as gradient descent. Although we do not need gradient descent to fit the model described above, we can still do so.

6. In gradient descent we minimise a loss function $L_\beta(\boldsymbol{y}, \boldsymbol{X})$ over the parameter values β. What is the loss function corresponding to the OLS description from before?
7. Assume we use a learning rate α. Write the update rule for a single gradient descent step on our parameters β.

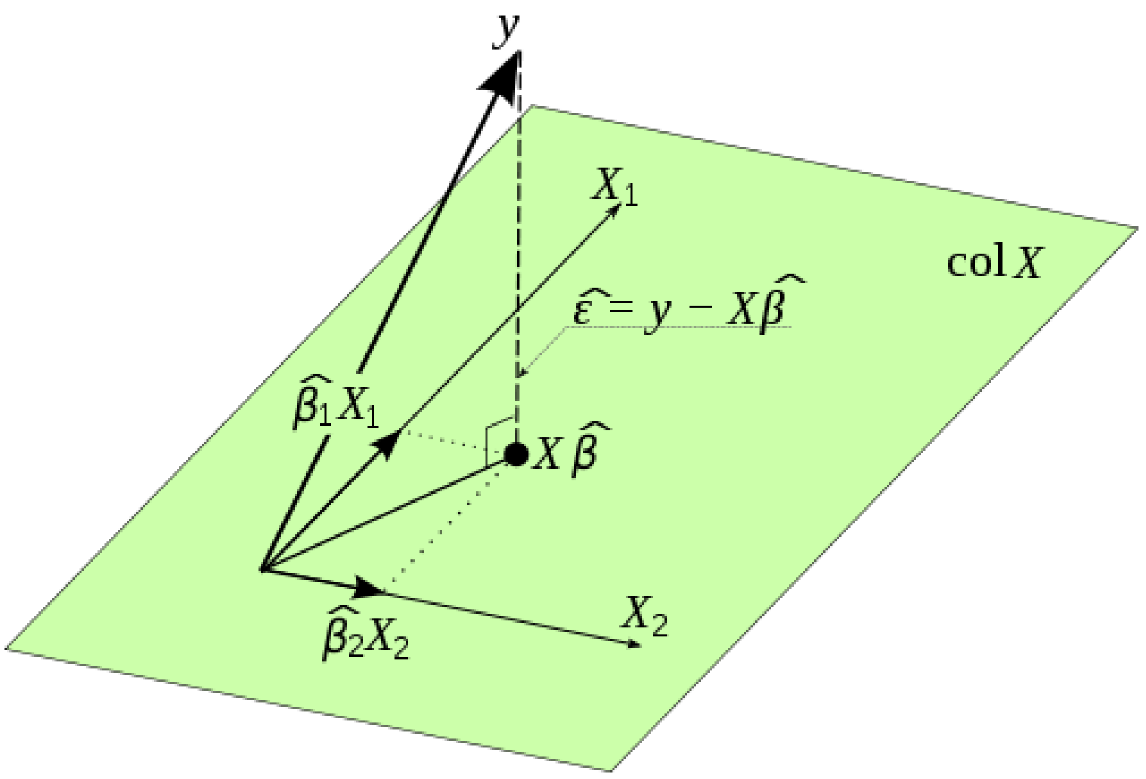

Figure 1: Geometric representation of OLS, from Wikipedia. The X_i are columns of $\boldsymbol{X}$.

Solution

1. The dimensionality of β is m.
2. We set $L = (\boldsymbol{y} - \boldsymbol{X}\beta)^2$, and set $\frac{\partial L}{\partial \beta} = 0$.

$$\begin{aligned} 0 &= \frac{\partial L}{\partial \beta} \\ 0 &= -2\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\beta) \\ \boldsymbol{X}^T\boldsymbol{y} &= \boldsymbol{X}^T\boldsymbol{X}\beta \\ (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} &= \beta \end{aligned}$$

3. Viewing the figure, we can imagine moving $\boldsymbol{X}\beta$ around and seeing what happens to the residual. In particular, we see that there is no way to move $\boldsymbol{X}\beta$ such that the residual is smaller than at the pictured $\boldsymbol{X}\hat{\beta}$, where the residual is exactly orthogonal to the column space of X. This is equivalent to saying $\boldsymbol{X}\hat{\beta}$ is the orthogonal projection $P(\boldsymbol{y})$ of $\boldsymbol{y}$ onto this subspace. Note that all the vectors in the figure are n-dimensional, and that col$X$ is an m-dimensional subspace spanned by m of these vectors.
4. We said that the residual $\boldsymbol{\epsilon}_\beta$ is orthogonal to the column space of X. This means that the inner product of the residual with any of the columns of $\boldsymbol{X}$ is zero, thus $\boldsymbol{X}^T\boldsymbol{\epsilon}_\beta = 0$. Equivalently we can write $\boldsymbol{\epsilon}_\beta^T\boldsymbol{X} = 0$.

5. Using that $\boldsymbol{\epsilon}_\beta = (\boldsymbol{y} - \boldsymbol{X}\beta)$:

$$\begin{aligned} 0 &= \boldsymbol{X}^T\boldsymbol{\epsilon}_\beta \\ 0 &= \boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\beta) \\ 0 &= \boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{X}\beta \\ \boldsymbol{X}^T\boldsymbol{X}\beta &= \boldsymbol{X}^T\boldsymbol{y} \\ \beta &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \end{aligned}$$

6. $L_\beta(\boldsymbol{y}, \boldsymbol{X}) = ||\boldsymbol{y} - f_\beta(\boldsymbol{X})||^2$, where $||...||^2$ is the squared L_2 norm.

7. We move in opposite direction of the gradient of our loss function (since we want to reduce loss), so:

$$\begin{aligned} \beta_{t+1} &= \beta_t - \alpha\frac{\partial L}{\partial \beta_t} \\ \beta_{t+1} &= \beta_t + 2\alpha\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\beta_t) \end{aligned}$$

Lecture 1: Introduction

1.1 Introduction

1. Explain what is meant by the 'curse of dimensionality'.

2. Suppose you are trying to design a predator agent that can learn to catch a randomly moving prey on a 5×5 ***toroidal*** grid. You have been given the (x, y)-coordinates of the predator and the (x, y)-coordinates of the prey to use as the state.

   (a) How many possible states are there in this naive approach?

   (b) There is a way of reducing the state space considerably a priori. Write down how you would adapt the given state representation to reduce the size of the state space. Note that you only care about a representation that you can use to solve the problem.

   (c) How many possible states are there now?

   (d) What is the advantage of doing this?

   (e) Consider the Tic-Tac-Toe example in Chapter 1.5 of the book. Here, too, we can exploit certain properties of the problem to reduce the size of the state space. Give an example of a property you can exploit.

3. Suppose you want to implement a Reinforcement Learning agent that learns to play Tic-Tac-Toe, as outlined in Chapter 1 of the book.

   (a) Which agent do you think would learn a better policy in the end: a greedy agent that always chooses the action it currently believes is best, or a non-greedy agent that sometimes tries new actions? Why?

4. Assume we start with an exploration rate of ϵ, meaning that whenever the agent chooses an action, it has a probability of ϵ to pick an action at random, and a probability of $1 - \epsilon$ to pick the greedy action. If we assume the environment has been sufficiently explored, we may want to reduce the amount of exploration after some time.

   (a) Write down how you would do this.

   (b) Does your method work if the opponent changes strategies? Why/why not? If not, provide suggestions on a heuristic that can adapt to changes in the opponent's strategy.

Solution

1. From the book: 'the computational requirements grow exponentially with the number of state variables'.

2. (a) 5^4

   (b) Instead of using both (x, y)-coordinates, use the distance between the predator and the prey

   (c) 5^2

   (d) Alleviating the curse of dimensionality by reducing the number of state variables

   (e) By exploiting rotation invariance in the value function.

3. (a) We expect the curious agent to perform better: it will be able to discover strategies that the greedy agent may miss.
4. (a) Straightforward: annealing ϵ over time.
   (b) Time-based: No. Heuristics that would work: TD-error, curiosity

1.2 Exploration

1. In ϵ-greedy action-selection for the case of n actions, what is the probability of selecting the greedy action?
2. Consider a 3-armed bandit problem with actions $1, 2, 3$. If we use ϵ-greedy action-selection, initialization at 0, and ***sample-average*** action-value estimates, which of the following sequence of actions are certain to be the result of exploration? $A_0 = 1, R_1 = -1, A_1 = 2, R_2 = 1, A_2 = 2, R_3 = -2, A_3 = 2, R_4 = 2, A_4 = 3, R_5 = 1$.
3. You are trying to find the optimal policy for a two-armed bandit. You try two approaches: in the pessimistic approach, you initialize all action-values at -5, and in the optimistic approach you initialize all action-values at +5. One arm gives a reward of +1, one arm gives a reward of -1. Using a greedy policy to choose actions, compute the resulting Q-values for both actions after three interactions with the environment. In case of a tie between two Q-values, break the tie at random. *Note*: the initialization is *not* a sample.
4. Which initialization leads to a higher (undiscounted) return? What if you had broken the tie differently?
5. Which initialization leads to a better estimation of the Q-values?
6. Explain why one of the two initialization methods is better for exploration.

Solution

1. We have probability $1 - \epsilon$ of selecting a greedy action, and ϵ of selecting a random action uniformly. Thus, each individual action has base probability of selection of $\frac{\epsilon}{n}$. The probability of selecting a greedy action is thus $1 - \epsilon + \frac{\epsilon}{n}$.
2. Start of Q-values: [0, 0, 0]. After $A_0 = 1$: [-1, 0, 0]. After $A_1 = 2$: [-1, 1, 0]. After $A_2 = 2$: [-1, -0.5, 0]. After $A_3 = 2$: [-1, 0.333, 0]. After $A_4 = 3$: [-1, 0.333, 1]. Actions that were non-greedy: A_3, A_4
3. (a) Start of Q-values: [5, 5]. After $A_0 = 1$: [1, 5]. After $A_1 = 2$: [1, -1] (these two can be flipped with the same end result). After $A_2 = 1$: [1, -1]. Total return: 1+-1+1=1
   (b) Start of Q-values: [-5, -5]. After $A_0 = 1$: [1, -5]. After $A_1 = 1$: [1, -5]. After $A_2 = 1$: [1, -5]. Total return: 1+1+1=3. If the tie is broken differently: $A_0 = 2$: [-5, -1]. $A_1 = 2$: [-5, -1]. $A_2 = 2$: [-5, -1]. Total return: -1+-1+-1=-3
4. If the tie is broken one way: pessimistic has higher return (3 vs 2). If it's broken the other way, optimistic has higher return (-3 vs 2).
5. The optimistic initialization
6. Looking for an answer along the lines that the optimistic one is better due to unexplored options having a very high value and thus higher chance of actually being selected.

Lecture 2: MDPs and dynamic programming

2.1 Markov Decision Processes

1. (a) For the first four examples outlined in Section 1.2 of the book, describe the state space, action space and reward signal.
   (b) Come up with an example of your own that you might model with an MDP. State the action space, state space and reward signal.
   (c) Come up with an example for a problem that you might have trouble solving with an MDP. Why doesn't this fit the framework?
   (d) In mazes, the agent's position is often seen as the state. However, the agent's position alone in not always a sufficient description. Come up with an example where the state consists of the agent's location and one or more other variables.
   (e) Consider the example in exercise 3.3 of the book. Why might you choose to view the actions as handling the accelerator, brake and steering wheel? What is the disadvantage of doing this?
   (f) Why might you choose to view the actions as choosing where to drive? What is the disadvantage of doing this?
   (g) Can you think of some way to combine both approaches?

2. (a) Eq. 3.8 in the book gives the discounted return for the continuous case. Write down the formula for the discounted return in the episodic case.
   (b) Show that $\sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}$ if $0 \leq \gamma < 1$. (Hint: if you're stuck, have a look at the Wikipedia page on *geometric series*)
   (c) Consider exercise 3.7 in the book. Why is there no improvement in the agent?
   (d) How would adding a discount factor of $\gamma < 1$ help solve this problem?
   (e) How might changing the reward function help solve this problem?

Solution

1. (a) i. State space: all possible configurations of chess board. Action space: all allowed moves of one piece at a time. Reward signal: win/lose/draw
      ii. State space: all possible configurations of parameters. Action space: change of parameters. Reward signal: marginal cost
      iii. State space: limb configurations. Action space: change angles of limbs. Reward signal: penalizes falling, reward acceleration
      iv. State space: battery level, location of charger. Action space: enter/not enter. Reward signal: penalize running out of battery, reward collecting trash
   (b) Up to the student, as long as there is a state space, action space and reward signal that make sense.
   (c) E.g. non-Markov states
   (d) For example, in Pacman the state consists of the agent's location, as well as the location of the ghosts and food pellets. In a driving tasks, the state consists of the agent's location as well as the state of other vehicles and context variables such as time of day, season, traffic light status. In a maze, the possession of a key might be another state variable next to the agent's location.

(e) This low-level representation allows you to learn *how* to drive. The disadvantage is that this approach makes it very hard to navigate anywhere, since our representation is too fine-grained.

(f) This high-level representation is much better suited to learning how to navigate somewhere, and to reach high-level goals such as *go to the supermarket*. The disadvantage is that we have to assume the agent already knows how to drive a car.

(g) Two-level policies, hierarchical RL.

2. (a)

$$G_t = \sum_{k=0}^{T-t} \gamma^k R_{t+k+1} \tag{2}$$

(b)

$$\begin{aligned}
\sum_{k=0}^{\infty} \gamma^k &= \lim_{n\to\infty} (1 + \gamma + \gamma^2 + \cdots + \gamma^n) && (3)\\
(1-\gamma)\sum_{k=0}^{\infty} \gamma^k &= \lim_{n\to\infty} (1-\gamma)(1 + \gamma + \gamma^2 + \cdots + \gamma^n) && (4)\\
&= \lim_{n\to\infty} \left((1-\gamma) + (\gamma - \gamma^2) + (\gamma^2 - \gamma^3) + \cdots + (\gamma^n - \gamma^{n+1})\right) && (5)\\
&= \lim_{n\to\infty} \left(1 - \gamma + \gamma - \gamma^2 + \gamma^2 - \gamma^3 + \cdots + \gamma^n - \gamma^{n+1}\right) && (6)\\
&= \lim_{n\to\infty} \left(1 - \gamma^{n+1}\right) && (7)\\
\sum_{k=0}^{\infty} \gamma^k &= \lim_{n\to\infty} \frac{1-\gamma^{n+1}}{1-\gamma} && (8)
\end{aligned}$$

If $\gamma < 1$, $\lim_{n\to\infty} \gamma^n \to 0$, so:

$$\sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma} \tag{9}$$

(c) The return is always $0 + 0 + \cdots + 1 = 1$, regardless of how many time steps the agent takes.

(d) By adding γ, the return is now γ^K, with K the number of time steps until the robot escapes. Since $\gamma < 1$, the return is bigger if the robot escapes faster.

(e) By changing the reward to give a small penalty of e.g. $R_t = -0.01$ on every time step, the return is now $G_t = -0.01(T-t)$, which means the return is bigger if the robot escapes faster.

2.2 Homework: Dynamic Programming

1. Write the value, $v^\pi(s)$, of a state s under policy π, in terms of π and $q^\pi(s,a)$. Write down both the stochastic and the deterministic policy case.

2. In Policy Iteration, we first evaluate the policy by computing its value function, and then update it using a Policy Improvement step. You will now change Policy Iteration as given on page 80 of the book to compute action-values. First give the new policy evaluation update in terms of $Q^{\pi}(s,a)$ instead of $V^{\pi}(s)$. That is, write an update for $Q^{\pi}(s,a)$ in terms of the current *action-value* function. Your answer should not contain a value function.

3. Now change the Policy Improvement update in terms of $Q^{\pi}(s,a)$ instead of $V^{\pi}(s)$. Note that Policy Improvement uses deterministic policies.

4. The Value Iteration update, given in the book in Eq. 4.10 can also be rewritten in terms of Q-values. Give the Q-value Iteration update.

Lecture 3: Monte Carlo methods

3.1 Homework: Monte Carlo

1. Consider an MDP with a single state s_0 that has a certain probability of transitioning back onto itself with a reward of 0, and will otherwise terminate with a reward of 5. Your agent has interacted with the environment and has gotten the following three trajectories: $[0, 0, 5], [0, 0, 0, 0, 5], [0, 0, 0, 5]$. Use $\gamma = 0.9$.
   (a) Estimate the value of s_0 using first-visit MC.
   (b) Estimate the value of s_0 using every-visit MC.
2. What is a disadvantage of using *ordinary importance sampling* in off-policy Monte Carlo?
3. What is a disadvantage of using *weighted importance sampling* in off-policy Monte Carlo?

Lecture 4: Temporal Difference methods

4.1 Temporal Difference Learning (Application)

Consider an undiscounted Markov Decision Process (MDP) with two states A and B, each with two possible actions 1 and 2, and a terminal state T with $V(T) = 0$. The transition and reward functions are unknown, but you have observed the following episode using a random policy:

- $A \xrightarrow[r_3=-3]{a_3=1} B \xrightarrow[r_4=4]{a_4=1} A \xrightarrow[r_5=-4]{a_5=2} A \xrightarrow[r_6=-3]{a_6=1} B \xrightarrow[r_7=1]{a_7=2} T$

where the the arrow ($\rightarrow$) indicates a transition and a_t and r_t take the values of the observed actions and rewards respectively.

1. What are the state(-action) value estimates $V(s)$ (or $Q(s,a)$) after observing the sample episode when applying:

   (a) TD(0) (1-step TD)

   (b) TD(2) (3-step TD)

   (c) SARSA

   (d) Q-learning

   where we initialize state values to 0 and use a learning rate $\alpha = 0.1$

Solution

(a) TD(0) (1-step TD):

$$\begin{aligned}
V(A) = V(B) \quad &= 0 \\
V(A) = 0 + 0.1 * (-3 + 0 - 0) \quad &= -0.3 \\
V(B) = 0 + 0.1 * (4 + (-0.3) - 0) \quad &= 0.37 \\
V(A) = -0.3 + 0.1 * (-4 + (-0.3) - (-0.3)) \quad &= -0.7 \\
V(A) = -0.7 + 0.1 * (-3 + 0.37 - (-0.7)) \quad &= -0.893 \\
V(B) = 0.37 + 0.1 * (1 + 0 - 0.37) \quad &= 0.433
\end{aligned}$$

Final:

$$\begin{aligned}
V(A) &= -0.893 \\
V(B) &= 0.433
\end{aligned}$$

(b) TD(2) (3-step TD):

$$\begin{aligned}
V(A) = V(B) \quad &= 0 \\
V(A) = 0 + 0.1 * ((-3 + 4 - 4) + 0 - 0) \quad &= -0.3 \\
V(B) = 0 + 0.1 * ((4 - 4 - 3) + 0 - 0) \quad &= -0.3 \\
V(A) = -0.3 + 0.1 * ((-4 - 3 + 1) + 0 - (-0.3)) \quad &= -0.87 \\
V(A) = -0.87 + 0.1 * ((-3 + 1) + 0 - (-0.87)) \quad &= -0.983 \\
V(B) = -0.3 + 0.1 * (1 + 0 - (-0.3)) \quad &= -0.17
\end{aligned}$$

Final:

$$V(A) = -0.983$$
$$V(B) = -0.17$$

(c) SARSA:

$$\begin{aligned}
Q(A,1) = Q(A,2) = Q(B,1) = Q(B,2) &= 0 \\
Q(A,1) = 0 + 0.1 * (-3 + 0 - 0) &= -0.3 \\
Q(B,1) = 0 + 0.1 * (4 + 0 - 0) &= 0.4 \\
Q(A,2) = 0 + 0.1 * (-4 + (-0.3) - 0) &= -0.43 \\
Q(A,1) = -0.3 + 0.1 * (-3 + 0 - (-0.3)) &= -0.57 \\
Q(B,2) = 0 + 0.1 * (1 + 0 - 0) &= 0.1
\end{aligned}$$

Final:

$$Q(A,1) = -0.57$$
$$Q(A,2) = -0.43$$
$$Q(B,1) = 0.4$$
$$Q(B,2) = 0.1$$

(d) Q-learning:

$$\begin{aligned}
Q(A,1) = Q(A,2) = Q(B,1) = Q(B,2) &= 0 \\
Q(A,1) = 0 + 0.1 * (-3 + 0 - 0) &= -0.3 \\
Q(B,1) = 0 + 0.1 * (4 + 0 - 0) &= 0.4 \\
Q(A,2) = 0 + 0.1 * (-4 + 0 - 0) &= -0.4 \\
Q(A,1) = -0.3 + 0.1 * (-3 + 0.4 - (-0.3)) &= -0.53 \\
Q(B,2) = 0 + 0.1 * (1 + 0 - 0) &= 0.1
\end{aligned}$$

Final:

$$Q(A,1) = -0.53$$
$$Q(A,2) = -0.4$$
$$Q(B,1) = 0.4$$
$$Q(B,2) = 0.1$$

2. Choose a deterministic policy that you think is better than the random policy given the data. Refer to any of the state(-action) value estimates to explain your reasoning.

Solution

We can use the state-action value estimates from Q-learning or SARSA to learn about the best actions per state. Since both algorithms agree, we choose

$$\pi_{student}(A) = 2$$
$$\pi_{student}(B) = 1.$$

In general, we would prefer to use the Q-learning estimates since it is off-policy, i.e. we would find the optimal (deterministic) policy after convergence. Remark: the state-action values might look quite different if we applied the algorithms repeatedly on the episode. Therefore, with proper argumentation other policies are possible.

3. Let π_{random} denote the random policy used so far and $\pi_{student}$ denote the new policy you proposed. Suppose you can draw new sample episodes indefinitely until convergence of the value estimates.

   (a) Discuss how do you expect the final value estimates to differ if you ran Q-Learning with π_{random} as compared to $\pi_{student}$.

   (b) What problems may arise with π_{random} or $\pi_{student}$ respectively?

   (c) Do you think using an ϵ-greedy policy as behavior policy would be beneficial? Explain why/why not?

   Solution

   (a) Since Q-learning is an off-policy algorithm, it will converge to the optimal policy provided each state-action pair is visited an infinite amount of times in the limit. π_{random} fulfills this property, however $\pi_{student}$ doesn't. Hence, under $\pi_{student}$ we only update $Q(A, 2)$ and $Q(B, 2)$ (not necessarily to $Q^*(A, 2)$, $Q^*(B, 2)$ though) whereas the values of $Q(A, 1)$ and $Q(B, 1)$ will remain unchanged. Under π_{random}, we will find $Q^*(s, a)\ \forall s, a$.

   (b) As already discussed in (i), $\pi_{student}$ does not update half of the state-action pairs at all and is not guaranteed to converge to the optimal Q-values. Even though, π_{random} converges to the optimal values, it might be inefficient, because it purely explores and does not exploit any knowledge about the Q-values.

   (c) ϵ-greedy is guaranteed to find the optimal policy under Q-learning and finds it faster than π_{random}. Thus, it is beneficial provided one it interested in finding the optimal policy in the least amount of episodes.

4.2 Contraction Mapping

Consider the following Theorem, from Csaba Szepesvari's *Algorithms for Reinforcement Learning*:

Theorem 1. (Banach's Fixed Point Theorem). Let V be a Banach space and $T : V \to V$ be a contraction mapping. Then T has a unique fixed point. Further, for any $v_0 \in V$, if $v_{n+1} = Tv_n$, then $v_n \to_{||\cdot||} v$, where v is the unique fixed point of T and the convergence is geometric:

$$||v_n - v|| \leq \gamma^n ||v_0 - v|| \tag{10}$$

1. (a) Consider the contraction mapping $T[x] := 1 + \frac{1}{3}x$. Find the fixed point of this mapping.

   (b) Consider the contraction mapping $(T[f])(s) := \frac{-1}{2} f(s) + g(s)$. Find the fixed point of this mapping in terms of $g(s)$.

2. Consider an MDP with finite state space $\mathcal{S}$, finite action space $\mathcal{A}$, and discount factor γ. Let V be the value function which gives the value per state, and $R(s, a)$ the expected

immediate reward for action a in state s and $Pr(s'|s,a)$ the probability of transitioning to state s' after taking action a in state s. Then the *Bellman optimality operator* B^* is given by

$$(B^*V)(s) = \max_a \left[R(s,a) + \gamma \sum_{s' \in \mathcal{S}} Pr(s'|s,a)V(s') \right] \tag{11}$$

(a) Show that

$$|\max_z f(z) - \max_z h(z)| \leq \max_z |f(z) - h(z)| \tag{12}$$

for arbitrary $f(z)$ and $h(z)$.

Let $(X, ||\cdot||)$ be a metric space. $T : X \to X$ is said to be a contraction if there exists a constant $0 \leq c < 1$ such that

$$||T(x) - T(y)|| \leq c||x - y|| \quad \forall x \in X, \forall y \in X \tag{13}$$

In our case, the metric space is $(\mathcal{B}(\mathcal{S}), ||\cdot||_\infty)$, where $\mathcal{B}(\mathcal{S})$ is the space of bounded functions on $\mathcal{S}$. $\mathcal{B}(\mathcal{S}) = \{V : \mathcal{S} \to \mathbb{R} : ||V||_\infty < +\infty\}$. $(\mathcal{B}(\mathcal{S}), ||\cdot||_\infty)$ is a normed vector space. Thus, the distance between value functions is given by the supremum norm $||V_1 - V_2||_\infty = \max_{s \in \mathcal{S}} |V_1(s) - V_2(s)|$, or the largest difference between state values.

(b) Show that B^* is a contraction mapping in supremum norm, i.e.

$$||(B^*V_1) - (B^*V_2)||_\infty \leq c||V_1 - V_2||_\infty \tag{14}$$

(c) Why is this an important result?

Solution

1. (a) The fixed point of T is given by $T(x) = x$, so:

$$x = T(x) \tag{15}$$
$$x = 1 + \frac{1}{3}x \tag{16}$$
$$x = 1.5 \tag{17}$$

(b) The fixed point of T is given by $(Tf)(s) = f(s)$, so:

$$f(s) = (Tf)(s) \tag{18}$$
$$f(s) = \frac{-1}{2}f(s) + g(s) \tag{19}$$
$$\frac{3}{2}f(s) = g(s) \tag{20}$$
$$f(s) = \frac{2}{3}g(s) \tag{21}$$

2. (a) The difference between the two maxima is smaller than or equal to maximizing the difference.

If $\max_z f(z) \geq \max_z h(z)$:

$$|\max_z f(z) - \max_z h(z)| = |f(z^*) - \max_z h(z)| \tag{22}$$
$$\leq |f(z^*) - h(z^*)| \tag{23}$$
$$\leq \max_z |f(z) - h(z)| \tag{24}$$

Note that the second step holds because $\max_z f(z) \geq \max_z h(z)$.

If $\max_z f(z) < \max_z h(z)$:

$$|\max_z f(z) - \max_z h(z)| = |\max_z f(z) - h(z^*)| \tag{25}$$
$$\leq |f(z^*) - h(z^*)| \tag{26}$$
$$\leq \max_z |f(z) - h(z)| \tag{27}$$

(b)

$$\max_s |(B^*V_1)(s) - (B^*V_2)(s)| = \max_s \left| \max_a \left(R(s,a) + \gamma \sum_{s'} Pr(s'|s,a)V_1(s') \right) \right. \tag{28}$$
$$\left. - \max_a \left(R(s,a) + \gamma \sum_{s'} Pr(s'|s,a)V_2(s') \right) \right| \tag{29}$$

Let s^* be the corresponding arg max, then we use the result from a) directly to get from 29 to 44:

$$\max_s \left| \max_a \left(R(s,a) + \gamma \sum_{s'} Pr(s'|s,a)V_1(s') \right) \right. \tag{30}$$
$$\left. - \max_a \left(R(s,a) + \gamma \sum_{s'} Pr(s'|s,a)V_2(s') \right) \right| \tag{31}$$
$$= \left| \max_a \left(R(s^*,a) + \gamma \sum_{s'} Pr(s'|s^*,a)V_1(s') \right) \right. \tag{32}$$
$$\left. - \max_a \left(R(s^*,a) + \gamma \sum_{s'} Pr(s'|s^*,a)V_2(s') \right) \right| \tag{33}$$
$$\leq \max_a \left| \left(R(s^*,a) + \gamma \sum_{s'} Pr(s'|s^*,a)V_1(s') \right) \right. \tag{34}$$
$$\left. - \left(R(s^*,a) + \gamma \sum_{s'} Pr(s'|s^*,a)V_2(s') \right) \right| \tag{35}$$
$$= \max_a \left| \gamma \sum_{s'} Pr(s'|s^*,a)V_1(s') - \gamma \sum_{s'} Pr(s'|s^*,a)V_2(s') \right| \tag{36}$$
$$= \max_a \left| \gamma \sum_{s'} Pr(s'|s^*,a) \left(V_1(s') - V_2(s') \right) \right| \tag{37}$$

Now, since $V_1(s') - V_2(s') \leq |V_1(s') - V_2(s')|$ and $\gamma > 0$ and $Pr(s'|s^*, a) \geq 0$, we can only increase the sum by taking the absolute value of every term. Since the sum of absolute values (weighted by positive weights) is positive, we can then remove the outer absolute value operator.

$$\max_a \left| \gamma \sum_{s'} Pr(s'|s^*, a) \left(V_1(s') - V_2(s')\right) \right| \tag{38}$$

$$\leq \max_a \left| \gamma \sum_{s'} Pr(s'|s^*, a) \left|V_1(s') - V_2(s')\right| \right| \tag{39}$$

$$= \max_a \gamma \sum_{s'} Pr(s'|s^*, a) \left|V_1(s') - V_2(s')\right| \tag{40}$$

$$\leq \max_a \gamma \sum_{s'} Pr(s'|s^*, a) \max_s \ |V_1(s) - V_2(s)| \tag{41}$$

$$= \max_a \gamma \cdot 1 \cdot \max_s \ |V_1(s) - V_2(s)| \tag{42}$$

$$= \gamma \max_s \ |V_1(s) - V_2(s)| \tag{43}$$

$$= \gamma ||V_1 - V_2||_\infty \tag{44}$$

We have proved that $||(B^*V_1) - (B^*V_2)||_\infty \leq \gamma ||V_1 - V_2||_\infty$ and thus B^* is a contraction mapping in supremum norm.

(c) It proves that using value-iteration, the value function converges to a fixed point. Additionally (proven in book that is referenced) this fixed point is the optimal value function since it is the solution of the Bellman optimality equation.

4.3 Temporal Difference Learning (Theory)

1. We can use Monte Carlo to get value estimates of a state with $V_M(S) = \frac{1}{M} \sum_{n=1}^{M} G_n(S)$ where $V_M(S)$ is the value estimate of state S after M visits of the state and $G_n(S)$ the return of an episode starting from S. Show that $V_M(S)$ can be written as the update rule $V_M(S) = V_{M-1}(S) + \alpha_M [G_M(S) - V_{M-1}(S)]$ and identify the learning rate α_M.

Solution

$$V_M(S) = \frac{1}{M} \sum_{n=1}^{M} G_n(S) \tag{45}$$

$$= \frac{1}{M}[G_M(S) + \frac{M-1}{M-1} \sum_{n=1}^{M-1} G_n(S)] \tag{46}$$

$$= \frac{1}{M}[G_M(S) + (M-1)V_{M-1}(S)] \tag{47}$$

$$= V_{M-1}(S) + \frac{1}{M}[G_M(S) - V_{M-1}(S)] \tag{48}$$

$$\alpha = \frac{1}{M} \tag{49}$$

2. Consider the TD-error

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t). \tag{50}$$

   (a) What is $\mathbb{E}[\delta_t | S_t = s]$ if δ_t uses the true state-value function V^π

   Solution

$$\begin{aligned}\mathbb{E}[\delta_t | S_t = s] &= \mathbb{E}[R_t + \gamma V^\pi(S_{t+1}) - V^\pi(S_t) | S_t = s] &(51)\\ &= \mathbb{E}[R_t + \gamma V^\pi(S_{t+1}) | S_t = s] - V^\pi(s) &(52)\\ &= V^\pi(s) - V^\pi(s) &(53)\\ &= 0 &(54)\end{aligned}$$

   where the step from (52) to (53) follows from the Bellman equation.

   (b) What is $\mathbb{E}[\delta_t | S_t = s, A_t = a]$ if δ_t uses the true state-value function V^π

   Solution

$$\begin{aligned}\mathbb{E}[\delta_t | S_t = s, A_t = a] &= \mathbb{E}[R_t + \gamma V^\pi(S_{t+1}) - V^\pi(S_t) | S_t = s, A_t = a] &(55)\\ &= \mathbb{E}[R_t + \gamma V^\pi(S_{t+1}) | S_t = s, A_t = a] - V^\pi(s) &(56)\\ &= Q^\pi(s, a) - V^\pi(s) &(57)\\ &= A(s, a) &(58)\end{aligned}$$

   where $A(s, a)$ is the advantage function (which will become important in later lectures).

3. The Monte-Carlo error can be written as the sum of TD-errors if the value estimates don't change (cf. equation 6.6 in the book) as

$$G_t - V(S_t) = \sum_{k=t}^{T-1} \gamma^{k-t} \delta_k. \tag{59}$$

   (a) Show that the n-step error as used in

$$V_{t+n}(S_t) = V_{t+n-1}(S_t) + \alpha[G_{t:t+n} - V_{t+n-1}(S_t)], \quad 0 \le t < T \tag{60}$$

   can also be written as a sum TD errors (assuming the value estimates don't change).

   Solution

$$G_{t:t+n} - V_{t+n-1}(S_t) = \tag{61}$$

$$= R_{t+1} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n}) - V_{t+n-1}(S_t) + \gamma V_{t+n-1}(S_{t+1}) - \gamma V_{t+n-1}(S_{t+1}) \tag{62}$$

$$= \delta_t + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n}) - \gamma V_{t+n-1}(S_{t+1}) \tag{63}$$

$$= \sum_{k=t}^{t+n-2} \gamma^{k-t} \delta_k + R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n}) + \gamma^{n-1} V_{t+n-1}(S_{t+n-1}) \tag{64}$$

$$= \sum_{k=t}^{t+n-1} \gamma^{k-t} \delta_k \tag{65}$$

4.4 Homework: Maximization Bias

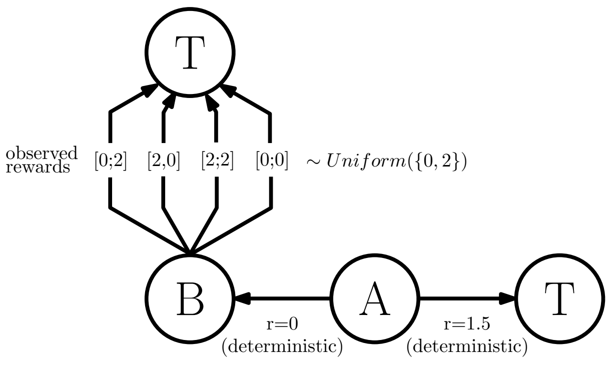

Figure 2: MDP: Maximization Bias

Consider the undiscounted MDP in Figure 2 where we have a state A with two actions, one ending in the terminal state T (with $V(T) = 0$) and always giving a reward of 1.5 and another action that transitions to state B with zero reward. From state B we can take a total of four different actions each transitioning to the terminal state and giving a reward of either 0 or 2 with equal probability. Suppose we sample all possible episodes two times and record the rewards. The observed rewards for each of the four actions from B are indicated in the Figure, e.g. the leftmost action received one time a reward of 0 and one time a reward of 2.

1. Suppose we repeatedly apply Q-learning and SARSA on the observed data until convergence. Give all final state-action values for Q-learning and SARSA respectively.
2. This problem suffers from maximization bias. Explain where this can be observed. Do both Q-learning and SARSA suffer from this bias? Why/why not?
3. To circumvent the issue of maximization bias, we can apply Double Q-learning. Use the given example to explain how Double Q-learning alleviates the problem of maximization bias.
4. What are the true state-action values that we would expect to get (after convergence) if we continued sampling episodes.