Not Completed

Principles of Machine Learning

Exercises

Victor Verreet *victor.verreet@cs.kuleuven.be*
Laurens Devos *laurens.devos@cs.kuleuven.be*

Fall, 2021

Exercise Session 8: Statistical Relational AI and Reinforcement Learning

8.1 Students and Grades

Four students have taken some courses and their grades are tabulated below. The table contains the ID of the student and the course. Every student implicitly has a skill level and every course has a difficulty. Grades are positively correlated with student skill and negatively correlated with course difficulty.

Student	1	2	1	2	3	4	3	4
Course	1	1	2	3	2	3	4	4
Grade	A	C	A	A	B	B	?	?

Which of the two students will then most likely have the best grade for the fourth course? Why? You do not need to perform an explicit calculation.

8.2 Weighted Model Counting

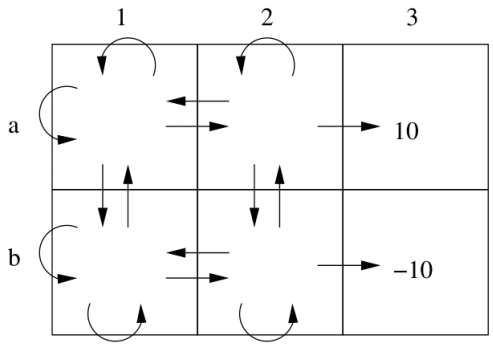
Behold the ProbLog program

$$0.6 :: a. \qquad d \leftarrow a, c.$$
$$0.2 :: b. \qquad d \leftarrow b.$$
$$0.3 :: c. \qquad e \leftarrow a, d.$$

which uses predicates a, b, c, d and e. A model is a truth assignment to these predicates such that the program is satisfied. Remember that ProbLog, like Prolog, follows the closed world semantics, meaning a predicate that cannot be proven from the facts is assumed to be false. What are then the models of this program? Calculate the probability that d is true by summing the weights of its models.

8.3 Reinforcement Learning in a Non-deterministic Environment

Consider the following environment

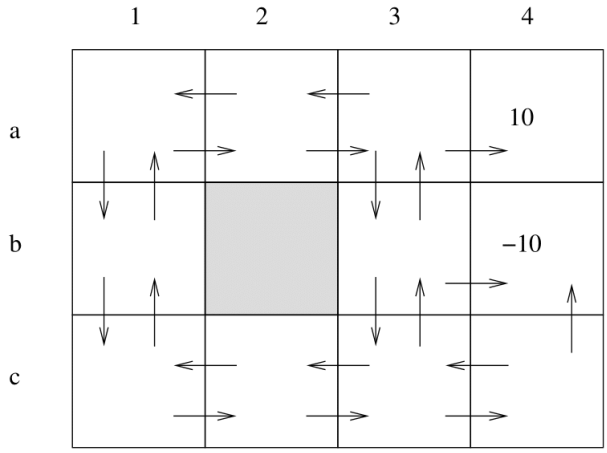


where entering the terminal states $A3$ and $B3$ gives a reward of 10 and a penalty of -10 respectively. At each position it is possible to move in each of the four directions. Bumping into a wall results in staying in the same position. Using a discount factor of $\gamma = 0.9$, write out the Bellman equations for the utility V^* of each state for the optimal policy

1. When actions are deterministic.
2. When actions are executed correctly in 70% of the cases, and another action is selected uniformly at random otherwise.

8.4 Iterative Policy Learning

Simulate Q-learning for a robot walking around in the following environment. $B2$ is a wall, entering $B4$ gives a penalty of -10 and entering $A4$ gives a reward of 10.



Indicate the Q-values after the following episodes, assuming the Q-values are initialized to 0. Use the back-propagated Q-update rule, meaning that after getting in a goal state, the Q-values are updated in reverse order from goal to start with $\gamma = 0.9$.

1. $C2$, $C1$, $B1$, $A1$, $A2$, $A3$, $A4$
2. $A1$, $A2$, $A3$, $B3$, $B4$
3. $C4$, $C3$, $B3$, $A3$, $A4$

Assume the robot will now use the policy of always performing the action having the greatest Q-value. Indicate this policy on the drawing. Is it optimal?

8.5 Continuous State Spaces and Generalization

Consider the following reinforcement learning problem. An agent is in a 10 meter long hallway. He starts at the leftmost end and receives a positive reward when he reaches the rightmost end. The position of the agent is described by a real number $x \in [0, 10]$ indicating its distance in meters from the leftmost end of the hallway, which is at $x = 0$. At each position, the agent has the option of taking

a step left or right. Since we are dealing with a continuous state-space it is not possible to use a straightforward table-based representation of the Q-function.

We have already estimated the Q-values for a couple of state-action pairs, which are given in the table below.

Example	State	Action	Q-Value
1	$x = 2$	$\leftarrow$	4.30
2	$x = 9$	$\rightarrow$	10.00
3	$x = 5$	$\rightarrow$	6.56
4	$x = 5$	$\leftarrow$	5.90

Given this data perform the following tasks.

1. Use linear regression to generalize the estimated values to all possible positions in the hallway.
2. Using these functions, compute and interpret the preferred action in both state $x = 8$ and state $x = 1$.

8.6 Simon Says

We want to teach an AI to play the game Simon Says. The game is played as follows: one person gives a finite sequence of movements, and the player has to exactly copy these. Let us for simplicity assume that there are only two possible movements, labelled 0 and 1. The states of this game are thus represented by finite strings containing only 0 and 1 that indicate which movements still have to be performed in order to win. The empty string then represents the winning state and let us also add one extra state $\perp$ to denote the failed state. The state space is then $\mathcal{S} = \{0, 1\}^* \cup \{\perp\}$. The possible actions are simply one of the two movements $\mathcal{A} = \{0, 1\}$. The transition function is then defined as follows

$$\tau : \mathcal{S} \times \mathcal{A} \to \mathcal{S} : (xy, a) \mapsto \begin{cases} y & \text{if } a = x \\ \perp & \text{otherwise} \end{cases}$$

in which $x \in \{0, 1\}$ represents the first character of the string and y is the rest. The transition function is not defined for the empty string and the failed state, because the game ends when these states are reached.

Notice that the state space is infinite and we thus cannot naively tabulate Q-values for all state-action pairs. Instead we will estimate Q-values by making a weighted average of all the observed Q-values. The weight of a Q-value is determined by the similarity of the states, as given by the Hamming distance

$$\delta(x_1 y_1, x_2 y_2) = \begin{cases} \delta(y_1, y_2) & \text{if } x_1 = x_2 \\ 1 + \delta(y_1, y_2) & \text{otherwise} \end{cases}$$

for $x_1 \in \{0, 1\}$ and $x_2 \in \{0, 1\}$. To compare strings of unequal length we also define $\delta(\epsilon, s) = \delta(s, \epsilon) = |s|$ with ϵ the empty string. The estimated Q-value for a state s is then

$$Q(s, a) = \frac{\sum_{s' \in D} Q(s', a) \delta(s, s')^{-1}}{\sum_{s' \in D} \delta(s, s')^{-1}}$$

where D is the set of states for which we know the Q-value.

If a training epoch consists of starting at a uniformly random chosen string of length n and then applying Q-learning, will the AI learn to correctly play the game for arbitrary starting strings in the limit of large data? Why?

8.7 Stopping Strategy

A fair six sided dice is rolled repeatedly and you observe outcomes sequentially. Formally the dice roll outcomes are independently and uniformly sampled from the set $\{1, 2, 3, 4, 5, 6\}$. At every time step before roll k you can choose between two actions:

1. Stop: stop and receive a reward equal to the number currently shown on the dice.
2. Roll: continue playing and receive no immediate reward.

If you do not stop before time step k, you are forced to take the Stop action, receive the corresponding reward and end the game.

The game state at time step n is represented by the number shown on the die at roll n. Assume that the discount factor γ is 1.

1. At time k the value function is $V_k(d) = d$ for $d \in \{1, 2, 3, 4, 5, 6\}$. Compute the value function at time $k - 1$.
2. Express the value function at time $n - 1$ recursively in terms of the value function at times n.
3. Notice that $Q_n(d, \text{Roll})$ does not depend on d, because the current roll gets discarded when choosing to reroll. We use the notation $q_n = Q_n(d, \text{Roll})$. Compute q_{k-1}.
4. Express q_{n-1} recursively as a function of q_n.
5. What is the optimal policy at roll n as a decision based on the current roll d and the value of q_n?

8.8 Value Iteration Convergence (Extra)

Given a finite state space $\mathcal{S}$ and a set of actions $\mathcal{A}$, the transition function $\tau : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is a map that defines the outcome $\tau(s, a)$ of taking action a in state s. Furthermore every state has a short-term reward function $r : \mathcal{S} \to \mathbb{R}$ that indicates how beneficial it is to be in state s. The value iteration algorithm tries to find the long-term value V of a state s assuming the optimal policy is followed. More precisely, V is a solution to the Bellman equation

$$V(s) = r(s) + \gamma \max_{a \in \mathcal{A}} V(\tau(s, a))$$

valid for all s, where $0 < \gamma < 1$ is the damping factor. In order to obtain V, we initialise $V^{(0)}(s)$ to some values and iteratively perform the update step

$$V^{(n)}(s) = r(s) + \gamma \max_{a \in \mathcal{A}} V^{(n-1)}(\tau(s, a))$$

until the changes

$$\Delta_n = \max_{s \in \mathcal{S}} \left| V^{(n)}(s) - V^{(n-1)}(s) \right|$$

are sufficiently small, say $\Delta_n \leq \epsilon$. In this exercise we will show that this update rule causes the algorithm to stop after a finite number of iterations. The values of $V^{(n)}$ will thus converge. To prove this, follow these steps.

1. Using the update equation, show that

$$\left| V^{(n+1)}(s) - V^{(n)}(s) \right| = \gamma \left| \max_{a \in \mathcal{A}} V^{(n)}(\tau(s, a)) - \max_{a \in \mathcal{A}} V^{(n-1)}(\tau(s, a)) \right|$$

   for all iterations n and all states s.

2. Argue that

$$\left| \max_{a \in \mathcal{A}} V^{(n)}(\tau(s, a)) - \max_{a \in \mathcal{A}} V^{(n-1)}(\tau(s, a)) \right| \leq \Delta_n$$

   for all iterations n and all states s. This is the most subtle step in the proof.

3. From the first two results derive that $\Delta_{n+1} \leq \gamma \Delta_n$.
4. Conclude that $\Delta_n \leq \epsilon$ for sufficiently large iteration number n. The algorithm will thus stop at this iteration.

8.3

① $V_{\pi^*}(s) = \sum_a \pi^*(a|s)\left(R_s^a + \gamma \sum P_{ss'}^a V_{\pi^*}(s)\right)$

$= \sum_a \pi^*(a|s)\left(R_s^a + 0.9\ V_{\pi^*}(s')\right)$

Deterministic

$B1 \to A1 \to A2 \to A3$

$B2 \to A2 \to A3$

$V_{\pi^*}(A1) = 0.9\ v^*(A2)$

$V_{\pi^*}(A2) = 10$

$V_{\pi^*}(B1) = 0.9 * v^*(A1)$

$V_{\pi^*}(B2) = 0.9\ v^*(A2)$

$v^*(A1) = 0.7\left(0.9 * (A2)\right) + 0.1\left(0.9 * v^*(B1)\right) +$

$0.2\left(0.9 * v^*(A1)\right)$

$v^*(A2) = 0.7\left(10\right) + 0.1\left(v^*(A1) \times 0.9\right) + 0.1\left(v^*(B2) \times 0.9\right)$

$+0.1\left(v^*(A2) \times 0.9\right)$

$v^*(B1) = 0.7\left(0.9 \times V_{\pi^*}(A1)\right) + 0.2\left(0.9 \times v^*(B1)\right) +$

$0.1\left(0.9 \times v^*(B2)\right)$

$V_{\pi^*}(B2) = 0.7\left(0.9 \times V_{\pi^*}(A2)\right) + 0.1\left(0.9 \times V_{\pi^*}(B1)\right) +$

$0.1(-10) + 0.1\left(0.9 \times V_{\pi^*}(B2)\right)$

$q^k(s,a) = q^{k-1}(s,a) + \alpha\left(R_s^a + \gamma \max_{a''} q(a', s_{t+1}) - q(a, s_t)\right)$

8.4

$q^{(1)}(A_3, \to) = 10$

$q^{(1)}(A_2, \to) = 0 + 1(10 \times 0.9 - 0) = 9$

$q^{(1)}(A1, \to) = 0 + 1(0 + 0.9 \times 9 - 0)$

$= 8.1$

$q^{(1)}(B1, \uparrow) = 1(0.9 \times 8.1) = 7.3$

$q^{(1)}(C1, \uparrow) = 0.9 \times 7.3 = 6.6$

$q^{(1)}(C2, \leftarrow) = 0.9 \times 6.6 = 5.94$

$q^{(2)}(B3, \to) = -10$

$q^{(1)}(A3, \downarrow) = q^{(1)}(A3, \downarrow) + (0.9 \times 0 - 0)$

$= 0$

$q^{(2)}(A2, \to) = q^{1}(A2, \to) + \frac{1}{2}\left(\max_a q(A3, a) - q^{1}(A2, \to)\right)$

$= 9$

2. $q^{(1)}(A1, \to) = q^{(1)}(A1, \to) + (0.9 \times 9 - 8.1)$

$= 8.1$

$q^{(1)}(A3, \to) = 10$

$q^{(1)}(B3, \uparrow) = q^{(2)}(B3, \uparrow) + (0.9 \times 10 - 0)$

$= 9$

$q^{(2)}(C3, \uparrow) = 8.1$

$q^{(2)}(C4, \leftarrow) = 0.9 \times 8.1 = 7.3$

8.5

① $\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}$

$\Rightarrow y = y_1 + \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x - x_1)$

$y = 4.3 + \frac{4.3 - 5.9}{-3} \cdot (x - 2)$

$\Rightarrow y_{\leftarrow} = 4.3 + \frac{1.6}{3}(x - 2)$

$y_{\to} = 10 + \frac{3.44}{4}(x - 9)$

$y_{\to} = 10 + \frac{3.44}{4}(-1)$

$= 10 - 0.86 = 9.14$

$y_{\leftarrow} = 4.3 + 3.2 = 7.5 < y_{\to}$

$y_{\leftarrow} = 4.3 - \frac{1.6}{3}$

$= \frac{12.9 - 1.6}{3} = \frac{11.3}{3} = 3.77$

$y_{\to} = 10 + \frac{3.44}{4} \cdot (-8)$

$= 10 - 6.88$

$= 3.12$