Symbolic Machine Learning - Written Test

May 31, 2023

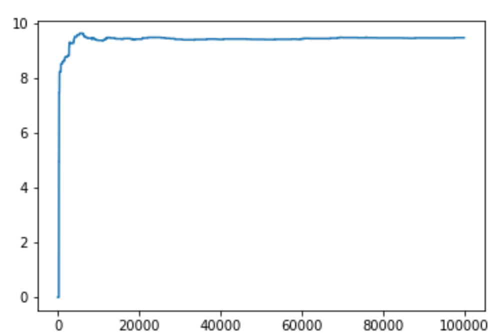
Your full name:

Use separate sheets for your answers unless you are asked to fill in a table directly. Do not fold the sheets, do not staple them together. Submit your signed sheets together with the signed assignment.

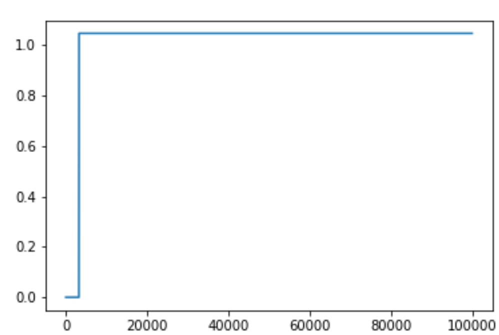
Question 1. (4 points)

Consider an active reinforcement learning algorithm. You are not told whether it is an instance of SARSA or Q-learning. The implementation met all convergence criteria. All plots shown below are related to the same state-action pair Q-value, i.e., $\widehat{Q}(s, a)$. The action a is **suboptimal** in state s. The used explore-exploit policy was the ε-greedy policy, i.e., with probability ε a random action is selected; otherwise, the agent behaves greedily.

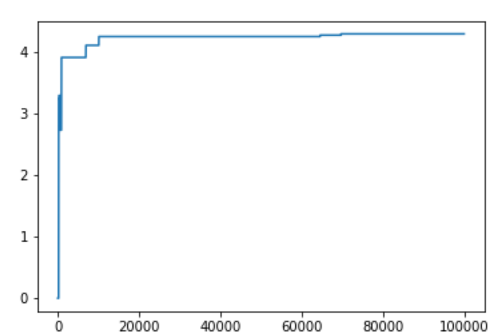
Now, consider four different situations of learning Q-values over 100 000 episodes. The plots show how estimates of $Q(s, a)$ change with the number of episodes, i.e., point $(50{,}000, 8.7)$ in the plot means that after 50,000 episodes, $\widehat{Q}(s, a)$ was 8.7.



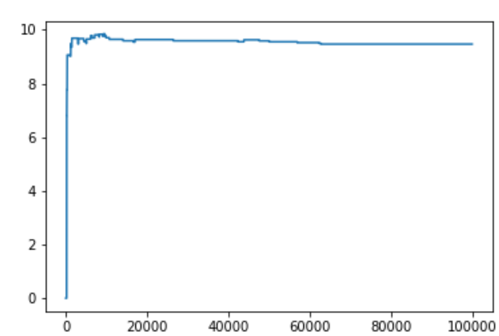
1.



2.

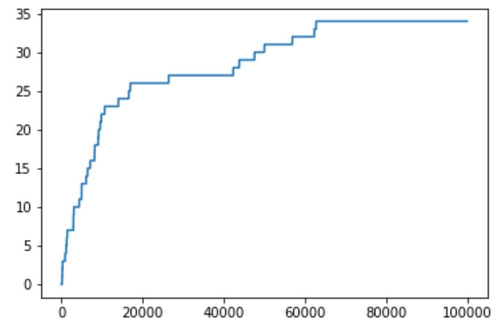


3.

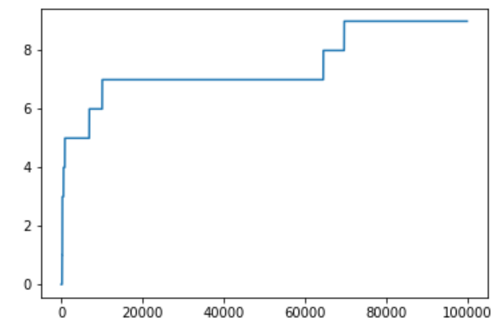


4.

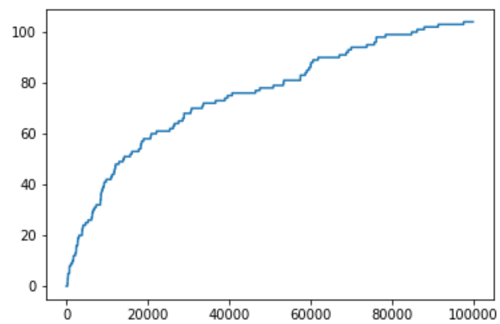
1. (2 points) The four plots below show how many times action a was selected by the agent in state s. For example, point $(1000, 6)$ means that the action a was selected 6 times in state s over the first 1000 episodes.



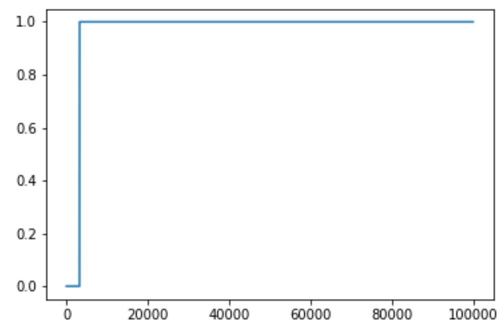
a.



b.



c.



d.

Match Figures a-d to Figures 1-4. Explain your decision.

2. (2 points) The ε was set as a function of the number of visits of state s. Relate the following four functions to Figures 1-4. Explain your choice.

$$\text{i.}\quad \varepsilon(n_s) = \frac{8}{7+n_s} \qquad \text{ii.}\quad \varepsilon(n_s) = \frac{3}{2+n_s} \qquad \text{iii.}\quad \varepsilon(n_s) = \frac{100}{99+n_s} \qquad \text{iv.}\quad \varepsilon(n_s) = \frac{1000}{999+n_s}$$

Answer:

Please see the tutorial solutions for an example of a correct answer and similar questions.

Question 2. (2 points)

Decide whether the following statement is true or false: *If a policy π is greedy with respect to its own value function V^π, then this policy is an optimal policy.* Briefly explain your answer; there is no need for a formal proof, one or two-sentence answer is enough.

Answer:

This problem was solved on the second tutorial. Please see the tutorial notes for an answer.

Question 3. (5 points)

Provide the update rule for the Q-learning and SARSA algorithms. Then, answer the questions below.

1. What is the difference between the on-policy and off-policy algorithms?

2. Decide whether the Q-learning and SARSA algorithms belong to the on-policy or off-policy cathegory.

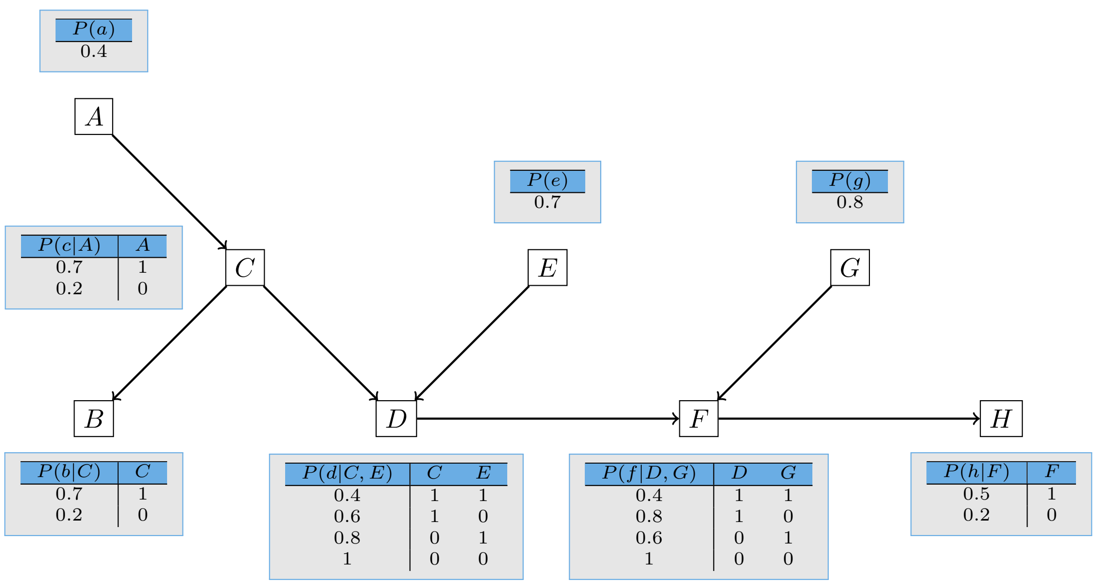
3. Which of these algorithms will converge to an optimal solution even in the presence of an adversarial man-in-the-middle between the agent and the environment, who occasionally alters the agent's chosen action? Assume that all other convergence criteria (GLIE, etc.) are met.

Answer:

The update rule and the answer to the first two questions can be found in lecture slides. Since Q-learning uses the max operator in the update rule, the Q-learning algorithm is independent of the action passed to the environment. Therefore, it will learn the Q-values of an optimial policy.

Question 4. (5 points)

Consider the network below. Calculate the probability $P(A = 1|B = 0, E = 1)$.



$P(a)$
0.4

$P(e)$
0.7

$P(g)$
0.8

$P(c\|A)$	A
0.7	1
0.2	0

$P(b\|C)$	C
0.7	1
0.2	0

$P(d\|C, E)$	C	E
0.4	1	1
0.6	1	0
0.8	0	1
1	0	0

$P(f\|D, G)$	D	G
0.4	1	1
0.8	1	0
0.6	0	1
1	0	0

$P(h\|F)$	F
0.5	1
0.2	0

Answer:

$$P(A = 1|B = 0, E = 1) = \frac{P(A = 1, B = 0, E = 1)}{P(B = 0, E = 1)}$$

To avoid several pages worth of computations, we need to find an appropriate elimination order:

$$P(B = 0, E = 1) = P(E = 1) \sum_a P(a) \sum_c P(c|a)P(B = 0|c) \underbrace{\sum_d P(d|c, E = 1) \sum_g P(g) \sum_f P(f|d, g) \sum_h P(h|f)}_{1}$$

Now, we can introduce a factor to hold results after eliminating C:

$$\mu_C(A = 0) = P(C = 0|A = 0)P(B = 0|C = 0) + P(C = 1|A = 0)P(B = 0|C = 1) = 0.8 \cdot 0.8 + 0.2 \cdot 0.3 = 0.7$$
$$\mu_C(A = 1) = P(C = 0|A = 1)P(B = 0|C = 0) + P(C = 1|A = 1)P(B = 0|C = 1) = 0.3 \cdot 0.8 + 0.7 \cdot 0.3 = 0.45$$

$$P(B = 0, E = 1) = P(E = 1) \sum_a P(a) \cdot \mu_C(a) = P(E = 1) \cdot (P(A = 0)\mu_C(A = 0) + P(A = 1)\mu_C(A = 1))$$

$$P(B = 0, E = 1) = 0.7 \cdot (0.6 \cdot 0.7 + 0.4 \cdot 0.45) = 0.7 \cdot (0.42 + 0.18) = 0.7 \cdot 0.6 = 0.42$$

Note that here, we can reuse the already computed factor:

$$P(A = 1, B = 0, E = 1) = P(E = 1) \cdot \mu_C(A = 1) = 0.7 \cdot 0.18 = 0.126$$

$$P(A=1|B=0,E=1) = \frac{0.126}{0.42} = \frac{0.7 \cdot 0.18}{0.7 \cdot 0.6} = 0.3$$

Question 5. (10 points)

In this question, assume the following corpus of 5 sentences:

1. *I love this test*
2. *I test this love*
3. *this test I love*
4. *this love I test*
5. *I love this*

For simplicity, denote the start *and* end of each sentence with *the same* special symbol $< s >$. Treat it like any other word.

1. (4 points) Create a Markov (bi-gram) probabilistic language model from the given corpus.
   It is sufficient to fill in only the values necessary to solve the sub-task 3.
2. (2 points) Apply Laplace (add-one) smoothing to the model and write down all the necessary calculations.
3. (2 points) Using the smoothed model, calculate probability of the sentence "I test this", including its start and end.
4. (1 points) Assume a sentence "I love this dog". How would you alter the model to assign some probability here?
5. (1 points) Assume comparing probabilities of much longer sentences. What problem could occur and how to solve it?

Question 6. (4 points)

1. (3 point) Describe the Skip-Gram with Negative Sampling (SGNS) variant of the word2vec algorithm.
2. (1 point) What is the training objective, learnable parameters, and hyperparameters?

Question 7. (2 points)

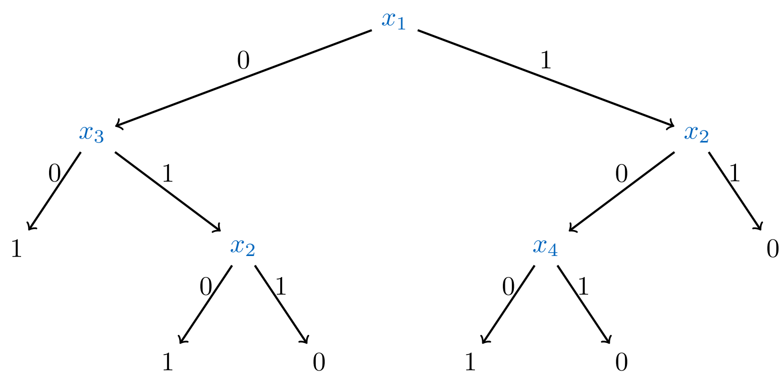
1. (1 point) What is (unsupervised) topic modeling in NLP?
2. (1 point) What are the inputs and outputs of such a topic model?

Question 8. (4 points)

1. (2 point) Using a diagram, describe a (high-level) structure of an LSTM cell.
2. (2 point) What are neural "gates"? What is the common idea and how is it implemented?

Question 9. (4 points)

Consider the following decision tree:



1. (1 point) Express the tree as a 3−DNF.

   Answer:

$$(\neg x_1 \wedge \neg x_3) \vee (\neg x_1 \wedge x_3 \wedge \neg x_2) \vee (x_1 \wedge \neg x_2 \wedge \neg x_4)$$

2. (1 point) Express the tree as a 3−CNF.

   Answer:

$$(x_1 \vee \neg x_3 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2)$$

3. (2 points) Can we use (modify) the generalization algorithm to learn $k-$decision trees in the mistake bound learning model?

   If we can, explain how. Will we also learn efficiently?

   If we cannot, explain why. Does it also mean that we cannot learn $k-$decision trees in the PAC learning model?

   Answer:

   Yes.

   We can use the fact following from the exercises above that any $k-$decision tree can be encoded as a $k-$CNF ($k-$DNF). Hence, we will use the generalization algorithm for learning $k-$CNFs ($k-$DNFs). Each clause (minterm) will be encoded by a new propositional variable and then we will simply be learning conjunction (disjuction) on literals. There will be $\sum_{i=0}^{k} \binom{n}{i} 2^i \leq poly(n)$ propositions, hence we will learn efficiently.

Question 10. (9 points)

Prove that the VC dimension of monotone conjunctions on n variables is n.

Answer:

- "At least n": we construct a set of n samples shattered by the monotone conjunctions, in particular:

$$\begin{array}{llllll} & x[1] & x[2] & x[3] & \dots & x[n] \\ x_1 = & 0 & 1 & 1 & \dots & 1 \\ x_2 = & 1 & 0 & 1 & \dots & 1 \\ x_3 = & 1 & 1 & 0 & \dots & 1 \\ \dots & & & & & \\ x_n = & 1 & 1 & 1 & \dots & 0 \end{array} \tag{1}$$

  With propositional variables $h_1, h_2, \dots h_n$, any subset $\{ x_i : i \in I \}$, $I \subseteq \{ 1, 2, \dots n \}$ of the above sample set is isolated with the monotone conjunction $\bigwedge_{i \notin I} h_i$. For example, the monotone conjunction $h_1 h_4 \dots h_n$ picks exactly the elements $\{ x_2, x_3 \}$. So the sample set is indeed shattered.

- "No more than n": to shatter a set of more than n elements, we need more than 2^n hypotheses but there are only 2^n monotone conjunctions on n variables.