Assignment10 – Learning

Given: July 10 Due: July 15

Problem 10.1 (Passive Reinforcement Learning)

Consider the example on Passive Learning in 4×3 world from the slides.

1. Give the transition model to the extent that it can be learned from these trials.

Solution: To obtain $P(s' \mid s, a)$, we count how often s' followed s after executing a. Concretely, we obtain

- $P((1,2) \mid (1,1), Up) = 2/3$ and $P((2,1) \mid (1,1), Up) = 1/3$
- $P((1,3) \mid (1,2), Up) = 3/3$
- $P((2,3) \mid (1,3), Right) = 2/3$ and $P((1,2) \mid (1,3), Right) = 1/3$
- $P((3,1) \mid (2,1), Left) = 1/1$
- $P((3,3) \mid (2,3), Right) = 2/2$
- $P((3,2) \mid (3,1), Left) = 1/1$
- $P((3,3) \mid (3,2), Up) = 1/2$ and $P((4,2) \mid (3,3), Up) = 1/2$
- $P((3,2) \mid (3,3), Right) = 1/3$ and $P((4,3) \mid (3,3), Right) = 2/3$

The transition probabilities for any action in state $(4,1)$ and for other action in the other states are not learned.

---

2. How could we learn the entire model?

Solution: Because we use a fixed policy, we always apply the same a whenever reaching s. So only some parts of the transition model can be learned from these trials. To learn the entire model, we have to try all actions in each state.

---

3. How would we proceed to learn the utilities of the states?

Solution: Once we have learned the transition model, we can apply the Bellman equation to obtain a system of linear equations in the utilities of each state and solve that. (If we are only interested in the utilities of states relative to a fixed policy, we only need the partial transition model learned from trials with that policy.)

Problem 10.2 (Active Reinforcement Learning)

Consider reinforcement learning in an unknown non-deterministic environment.

1. Explain the difference between a passive and an active agent.

2. What is the critical trade-off in designing an actively learning agent?

Solution:

1. A passive agent has a fixed policy and is only trying to learn its utility. An active agent is additionally trying to find the best policy.
2. Because the optimal policy depends on the transition model, the agent must first learn the transition model and then find the optimal policy. But it is difficult to decide when to switch from the former to the latter. If it switches too early, the transition model has not been learned well yet and (garbage in, garbage out) the computed policy is bad. If it switches too late, it wastes time and resources learning the transition model in areas that are not visited by the optimal policy anyway.

Problem 10.3 (Logical Formulation of Learning)

Some people with different attributes go sunbathing, the result class is whether they get sunburned:

#	Hair	Height	Weight	Lotion	Sunburned
1	Blonde	Short	Light	No	Yes
2	Blonde	Short	Average	Yes	No
3	Brown	Short	Average	Yes	No
4	Blonde	Short	Average	No	Yes
5	Blonde	Tall	Heavy	No	Yes
6	Brown	Tall	Heavy	No	No

We want to model these examples in first-order logic.

1. Explain which predicates are needed. For each predicate give the arity and the domain of each argument. Give the formal representation of the description and the classification of Example 1.

Solution: We use a unary predicate for each boolean attribute/class and a binary predicate for the others. The first argument of all predicates represents the example and has domain $\{1, \dots, 6\}$. The domain of the second argument of the binary predicates is the set of possible values. Thus, we have for an example e:

- $Hair(e, c)$ for $c \in \{Blonde, Brown\}$
- $Height(e, h)$ for $h \in \{Short, Tall\}$
- $Weight(e, w)$ for $w \in \{Light, Average, Heavy\}$
- $Lotion(e)$
- $Sunburned(e)$

For exampel, the description of example 1 is $Hair(1, Blonde) \wedge Height(1, Short) \wedge Weight(1, Light) \wedge \neg Lotion(1)$. Its classification is $Sunburned(1)$.

---

2. For attributes A, B, we write $A_1, \ldots, A_n \succ B$ if any two examples that agree on all of the attributes A_i also agree on the attribute B.
Explain whether Height$\succ$ Weight and Weight $\succ$ Height hold or do not hold.

Solution: Height $\succ$ Weight does not hold: Examples 1 and 2 agree on Height but not on Weight. Weight $\succ$ Height holds: any pair of examples that agree on Weight also agree on Height.

---

3. Give a minimal subset $\mathcal{A} \subseteq \{Hair, Height, Weight, Lotion\}$ such that $\mathcal{A} \succ Sunburned$ holds.
Using the predicates introduced above, give a logical formula that captures the rule $\mathcal{A} \succ Sunburned$.

Solution: $\mathcal{A} = \{Hair, Lotion\}$. This corresponds to the formula $\forall e, e'.\forall c.\big((Hair(e,c) \wedge Hair(e',c)) \wedge ((Lotion(e) \Leftrightarrow Lotion(e'))\big) \Rightarrow ((Sunburned(e) \Leftrightarrow Sunburned(e')))$.
For example, instantiating with $e' = 1$ and $c = Blonde$, that implies $\forall e.Hair(e, Blonde) \wedge \neg Lotion(e) \Rightarrow Sunburned(e)$.

Problem 10.4 (Inductive Learning)
Consider the family tree given by the following relations:

couple	children
A, B	E, F
C, D	G, H
F, G	I, J

Assume we already know the predicate $\texttt{par}(x, y)$ for x being a parent of y.
Our goal is to learn the predicate $\texttt{gp}(x, y)$ for x being a grandparent of y. That means to find a formula D such that $\forall x, y.\texttt{gp}(x, y) \Leftrightarrow D(x, y)$.
We do not know D, but we have the following examples for gp:

person-pair	grandparent
A, I	yes
B, I	yes
A, J	yes
A, E	no
A, F	no
F, A	no
A, A	no
C, J	yes
D, H	no
I, A	no

1. Give the intended formula D_1, i.e., the correct definition of grandparent.
2. Give a formula D_2 that covers exactly the positive examples.
3. Explain the pros and cons of learning the formula D as D_1 vs. D_2.
4. We want to learn algorithmically the formula $D(x, y) = \exists u_1, \ldots, u_l . L_1 \wedge \ldots \wedge L_k$ where each L_i is a literal of the form $P(x, y, u_1, \ldots, u_l)$ or $\neg P(x, y, u_1, \ldots, u_l)$ for some predicate symbol P including the equality predicate, i.e., $P \in \{\texttt{par}, \texttt{gp}, =\}$. We do so by building the set $\{L_1, \ldots, L_k\}$ of literals gradually (with the understanding that each L_i can have any free variables in addition to x and y, which we will collect at the end as the u_i).
   (a) If we start with the empty set of literals, give all useful choices that we can make for the first literal.
   (b) For each choice L that is non-recursive (i.e., P is not the target predicate gp), positive (i.e., the literal is not negated), not an equality (i.e., P is not the equality predicate $=$), and does not introduce a new variable (i.e., only uses x and y), which examples are falsely classified?

Solution:

1. $D_1(x, y) = \exists u . \texttt{par}(x, u) \wedge \texttt{par}(u, y)$
2. $D_2(x, y) = (x = A \wedge y = I) \vee (x = B \wedge y = I) \vee$ and so on for all positive examples.
3. pros of D_1:
   - D_2 captures only the provided examples. If that list is incomplete, it is likely the wrong formula. Even if the list is complete, D would have to be changed every time the list of examples changes. D_1 captures all future examples correctly.
   - D_1 has size $O(1)$, whereas D_2 has size $O(n)$ where n is the number of examples.

   pros of D_2:
   - D_2 can be easily read off the list of examples in $O(n)$ time.
   - A nice formula for D_1 might not even exist or might be very difficult to find or might be impossible to find from the given examples.
4. (a) There are 16 choices each for $P \in \{\texttt{par}, \texttt{gp}\}$: $P(x, x)$, $P(y, y)$ $P(x, y)$ or $P(y, x)$ (no new variables), $P(x, u)$, $P(u, x)$, $P(y, u)$, $P(u, y)$ (1 new variable); as well as the negated versions. (It is useless to add $P(u, v)$ or $P(u, u)$ where all variables are new.)
   Additionally, we can choose $x = y$ and $\neg x = y$. (It is useless to add an (in)equality with a new variable or where both variables are the same.)
   (b) There are 4 such choices:
   - $\texttt{par}(x, x)$: no false-positives
   - $\texttt{par}(y, y)$: no false-positives
   - $\texttt{par}(x, y)$: false-positives are (A, E), (A, F), and (D, H)
   - $\texttt{par}(y, x)$: false-positives are (F, A)

   Additionally, for all 4 choices, all positive examples are false-negatives.
