CSE 241 Artificial Intelligence
End-Semester Assignment

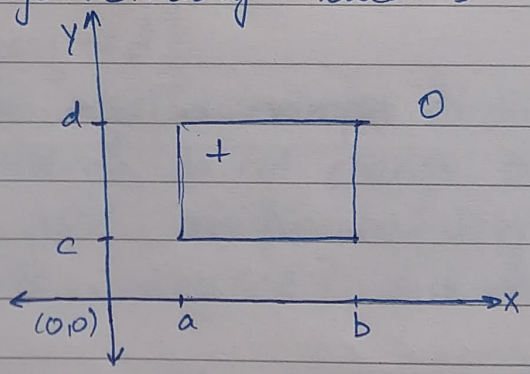
Enroll
Page No:
Date: / /

Name:- Shruti T. Avhad
Roll No.: 20074030
Branch:- CSE (IDD)
Email:- shruti.tavhad.cse20@itbhu.ac.in

Ans 1.) Given the instance space consisting of integer points in the x, y plane. The set of hypothesis H consists of rectangles of the form $a \le x \le b$, $c \le y \le d$, where a, b, c, d are integers.

a) Let the instance space be denoted by X, and the target concept be c. Instances of training set, where $c(x)=1$ are called positive examples and $c(x)=0$ are negative examples.
The hypothesis H can be geometrically viewed as -

$H = \{ a \le x \le b,\ c \le y \le d \}$



let the points lying inside or on the rectangle be the positive examples (denoted by '+') and those lying outside the rectangle be negative examples (denoted by '0').
Now, we define the most general 'G' and most specific 's' boundaries wrt hypothesis space H, that completely specify the version space.
$G = \{ -\infty < x < \infty,\ -\infty < y < \infty \}$, i.e the complete x, y cartesian plane.
We use Find-S algorithm to determine 's' boundary.

1. Initializing h to the most specific hypothesis in H, i.e, $h = \{ \phi, \phi \}$, where ϕ indicates no instance from the instance space is acceptable.

Teacher's Signature

Enroll
Page No:
Date: / /

2. We generalize h each time it fails to cover / correctly classify a positive training example.
A positive example is $(a <= x <= a, c <= y <= c)$ which is also the positive example with min. values of x and y. Replacing h by the most general constraint that fits this example.

$h = \{ a <= x <= a, c <= y <= c \}$.

3. Still h is very specific, next we generalize it to cover the second positive instance $(b <= x <= b, d <= y <= d)$ which is the positive training example with max. value of x and y.

$h = \{ a <= x <= b, c <= y <= d \}$.

Since, Find-S algo ignores negative examples so the training instances outside the rectangle do not change the hypothesis h. Thus, the most specific boundary s is given by $s = \{ a <= x <= b, c <= y <= d \}$ and the most general boundary G is

$G = \{ -\infty < x < \infty, -\infty < y < \infty \}$

Initial S, that is most specific hypotheses will be empty / null set

$S_0 : \{ < \phi, \phi, \phi, \phi > \}$

Enroll
Page No:
Date: / /

(b) Let the G_1 and S boundaries that we currently have been defined as -

$$G_1 : \{< a_{G_1}, b_{G_1}, c_{G_1}, d_{G_1} >\}$$
$$S : \{< a_S, b_S, c_S, d_S >\}$$

If we take a positive Training Example (x,y) such that -

$x < a_S$ or $x > b_S$ or $y < c_S$ or $y > d_S$

i.e (x,y) is outside the rectangle specified by S, then S will be generalised to be consistent with this training example and this will reduce the size of the version space.

but if (x,y) is within or on the rectangle specified by S i.e. $a_S \leq x \leq b_S$ and $c_S \leq y \leq d_S$ then this training example will not cause any change to the version space.

→ If we take a Negative Training Example (x,y) such that -

$a_{G_1} \leq x \leq b_{G_1}$ and $c_{G_1} \leq y \leq d_{G_1}$

i.e (x,y) is inside (or on) the rectangle specified by G_1, then G_1 will be specialized to be consistent with this training example and this will reduce the size of version space but if (x,y) is outside the rectangle specified by G_1, then it will not affect the version space.

Enroll
Page No:
Date: / /

c) For each positive training example, we generalize S and for each negative training example, we make G more specific.

We need 2 positive training examples to learn S
and 4 negative training examples to learn G.
∴ Total 6 training examples are needed (min.) to learn target concept perfectly.

Eg: 2 positive training examples

S: {<ϕ, ϕ, ϕ, ϕ>} (a, c) - positive ex. with smallest x,y coord.

↓

S: {<a, a, c, c>} (b, d) - +ve ex. with largest x, y coord.

↓

S: {<a, b, c, d>}

4 negative training examples

G_1: {<−∞, ∞, −∞, ∞>}

↓

1.) ((a+b)/2, c−1) G_1: {<−∞, ∞, c, ∞>}

↓

2.) ((a+b)/2, d+1) G_1: {<−∞, ∞, c, d>}

↓

3.) (a−1, (c+d)/2) G_1: {<a, ∞, c, d>}

↓

4. (b+1, (c+d)/2) G_1: {<a, b, c, d>}

Enroll
Page No:
Date: / /

Q2.)

Ans- CANDIDATE - ELIMINATION ALGORITHM

Initially,

$G_0 \leftarrow \{ <?, ?, ?, ?, ?, ?> \}$ (most general hypothesis)

$S_0 \leftarrow \{ <\phi, \phi, \phi, \phi, \phi, \phi> \}$ (most specific hypothesis)

Instance 1:

< Sunny, Warm, Normal, Strong, Warm, Same >

EnjoySport: Yes (positive example)

$S_0 : \{ <\phi, \phi, \phi, \phi, \phi, \phi> \}$

↓

$S_1 : \{ <$ Sunny, Warm, Normal, Strong, Warm, Same $> \}$

$G_0, G_1 : \{ <?, ?, ?, ?, ?, ?> \}$

Instance 2:

< Sunny, Warm, High, Normal, Warm, Same >

EnjoySport: Yes (positive Example)

$S_1 : \{ <$ Sunny, Warm, Normal, Strong, Warm, Same $> \}$

↓

$S_2 : \{ <$ Sunny, Warm, ?, ?, Warm, Same $> \}$

$G_1, G_2 : \{ <?, ?, ?, ?, ?, ?> \}$

Enroll

Page No:

Date: / /

Instance 3:

<Rainy, Cold, High, Strong, Warm, Change>

Enjoy Sport : No (negative Example)

S_2, S_3 : {<Sunny, Warm, ?, ?, Warm, Same>}

G_3 : {<Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?>, <?, ?, ?, ?, ?, Same>}

↑

G_2 : {<?, ?, ?, ?, ?, ?>}

Instance 4:

<Rainy, Warm, High, Strong, Cool, Change>

Enjoy Sport : No (negative Example)

S_3, S_4 : {<Sunny, Warm, ?, ?, Warm, Same>}

G_4 : {<Sunny, ?, ?, ?, ?, ?>, <?, ?, ?, ?, ?, Same>}

↑

G_3 : {<Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?>, <?, ?, ?, ?, ?, Same>}

∴

S_4 : {<Sunny, Warm, ?, ?, Warm, Same>}

G_4 : {<Sunny, ?, ?, ?, ?, ?>, <?, ?, ?, ?, ?, Same>}

Enroll
Page No:
Date: / /

Final version space:

S: {<sunny, Warm, ?, ?, Warm, same>}

↑

{<sunny, Warm, ?, ?, ?, ?>, <sunny, ?, ?, ?, Warm, ?>,
<sunny, ?, ?, ?, ?, same>, <sunny, Warm, ?, ?, Warm, ?>,
<sunny, Warm, ?, ?, ?, same>, <sunny, ?, ?, ?, Warm, same>,
<?, Warm, ?, ?, ?, same>, <?, ?, ?, ?, Warm, same>,
<?, Warm, ?, ?, Warm, same>}

↑

G1: {<sunny, ?, ?, ?, ?, ?>, <?, ?, ?, ?, ?, same>}

Enroll

Page No:

Date: / /

Q3.)

Ans- The criteria we use in Regression trees for binary recursive splitting is Residual sum of squares (RSS).

RSS measures the level of variance in the error term of the regression tree. It is defined as the sum of squares of difference of the true/target label of an observation within j^{th} box and the mean response of all training observations in that box/region. Mathematically,

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

y_i - True/target response of i^{th} observation in j^{th} box/region

$\hat{y}_{R_j}$ - Mean of responses of all training observations within the j^{th} box/region.

R_j - j^{th} Region

J - Total number of regions.

In binary recursive splitting for regression trees, a greedy top-down approach is taken where at each step, such a split is made that minimizes the RSS.

The criteria we use in Classification Trees for binary recursive splitting is Classification Error Rate (CE).

Classification Error Rate is the fraction of training observations in that region that do not belong to the most common class.

Mathematically,

$$E = 1 - \max_k (\hat{p}_{mk})$$

where $\hat{p}_{mk}$ - portion of training observations in the m^{th} region that are from k^{th} class.

Enroll

Page No:

Date: / /

Apart from classification Error Rate, which is insufficiently sensitive for tree-growing, two other criterion are also used.

1.) Gini Index

It is a measure of total variance across the K classes.

$$G_I = \sum_{K=1}^{K} \hat{p}_{mK} (1 - \hat{p}_{mK})$$

It is referred to as measure of node purity because if all of the $\hat{p}_{mK}$'s are close to 0 or 1, then G_I will take a small value, indicating that a node contains predominantly observations from a single class.

2.) Cross Entropy

It is quite similar to Gini Index numerically.

$$D = -\sum_{K=1}^{K} \hat{p}_{mK} \cdot \log \hat{p}_{mK}$$

Cross Entropy will take a small value (close to 0) if the $\hat{p}_{mK}$'s are all near 0 or near 1.

We cannot use the criteria for Regression that is RSS, in a classification tree because RSS is meant for measuring error between continuous and real-valued target labels and continuous real-valued predictions. since, in classification, the target labels and the prediction, both are categorical in nature, thus, RSS won't be a suitable choice. Same goes for Gini Index and Cross-Entropy. They are designed for categorical labels and predictions, and therefore cannot be used for continuous real-valued data as in the case of Regression.

Enroll

Page No:

Date: / /

Q4.)

Ans.

$X = \{1, 2, 3, 4\}$

$A = \left\{ \frac{0.1}{1} + \frac{0.25}{2} + \frac{0.55}{2} + \frac{0.9}{4} \right\}$

$B = \left\{ \frac{0.2}{1} + \frac{0.3}{2} + \frac{0.4}{3} + \frac{0.7}{4} \right\}$

$\therefore \mu_A = \{0.1, 0.25, 0.55, 0.9\}$

$\mu_B = \{0.2, 0.3, 0.4, 0.7\}$

1) Union

$\rightarrow A \cup B$

$\mu_{A \cup B} = \max(\mu_A, \mu_B)$

$= \{\max(0.1, 0.2), \max(0.25, 0.3), \max(0.55, 0.4), \max(0.9, 0.7)\}$

$= \{0.2, 0.3, 0.55, 0.9\}$

$\therefore \boxed{A \cup B = \left\{ \frac{0.2}{1} + \frac{0.3}{2} + \frac{0.55}{3} + \frac{0.9}{4} \right\}}$

2) Intersection

$\mu_{A \cap B} = \min(\mu_A, \mu_B)$

$= \{\min(0.1, 0.2), \min(0.25, 0.3), \min(0.55, 0.4), \min(0.9, 0.7)\}$

$= \{0.1, 0.25, 0.4, 0.7\}$

$\therefore \boxed{A \cap B = \left\{ \frac{0.1}{1} + \frac{0.25}{2} + \frac{0.4}{3} + \frac{0.7}{4} \right\}}$

3) Difference

$\mu_{\bar{A}} = 1 - \mu_A$

$= \{1 - 0.1, 1 - 0.25, 1 - 0.55, 1 - 0.9\}$

$= \underline{\{0.9, 0.75, 0.45, 0.1\}}$

Enroll

Page No:

Date: / /

$$\mu_{\bar{B}} = 1 - \mu_B$$
$$= \{1-0.2,\ 1-0.3,\ 1-0.4,\ 1-0.7\}$$
$$= \underline{\{0.8, 0.7, 0.6, 0.3\}}$$

$$A - B = A \cap \bar{B}$$

now, $\mu_{A\cap\bar{B}} = \min(\mu_A, \mu_{\bar{B}})$

$$= \{\min(0.1, 0.8), \min(0.25, 0.7), \min(0.55, 0.6), \min(0.9, 0.3)\}$$

$$= \underline{\{0.1, 0.25, 0.55, 0.3\}}$$

Similarly,

$$B - A = B \cap \bar{A}$$

$$\therefore \mu_{B\cap\bar{A}} = \min(\mu_B, \mu_{\bar{A}})$$
$$= \{\min(0.2, 0.9), \min(0.3, 0.75), \min(0.4, 0.45), \min(0.7, 0.1)\}$$
$$= \underline{\{0.2, 0.3, 0.4, 0.1\}}$$

$\therefore$

$$A - B = \left\{\frac{0.1}{1} + \frac{0.25}{2} + \frac{0.55}{3} + \frac{0.3}{4}\right\}$$

$$B - A = \left\{\frac{0.2}{1} + \frac{0.3}{2} + \frac{0.4}{3} + \frac{0.1}{4}\right\}$$

Enroll
Page No:
Date: / /

Q6.)

Ans - UCB Algorithm uses the explore-exploit balanced strategy to find the most optimal solution. The algorithm initially focuses on exploration, giving preference to actions that have been tried the least, and gradually concentrates on exploitation, selecting the action with the highest estimated reward.

In UCB, the action 'A_t' chosen at time step 't' is given by -

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

where, $Q_t(a)$ - estimated value of action 'a' at time step 't'.

$N_t(a)$ - No. of times action 'a' has been selected prior to time 't'.

c - confidence value.

The two terms that govern the selection of an action in UCB algo are:

1) Exploitation:

- The $Q_t(a)$ term represents the exploitation part of the equation. This term favours the action that has the highest estimated reward at that time step.

2) Exploration:

- The second half term of above equation controls the exploration part. The confidence value 'c' controls the degree of exploration. The $N_t(a)$ term in the denominator will influence the selection of an action depending on the no. of times it has already been selected. If $N_t(a)$ is small, the uncertainty (exploration term will be large, making

Enroll
Page No:
Date: / /

this action more likely to be selected. If $N_t(a)$ is large, the uncertainty term will be small, making it less likely for the action to be selected because of exploration.

The parameter 'c' denotes the confidence value that controls the level of exploration. A larger 'c' will make the uncertainty / exploration term in the above equation large which will favour the exploration of the action.

When we use optimistic initial values, we assume that all actions are optimum at first. This is done by assigning all actions an initial value larger than mean reward we expect to get. As a result of this, the first estimate of any action will always be less than this optimal initial value. And, even if we're greedy, we'll not visit this action again and rather select another unvisited action.

∴ Advantage of using optimistic initial values is that it encourages the agent to visit all actions multiple times, resulting in early improvement in estimated action values.

Enroll
Page No:
Date: / /

Q7.)

Ans- The problem with the given solution is that it is suffering from exploration issue where the agent is unable to find the exit first time and therefore doesn't know if there's anything better than 0 as a reward.

The communication is not effective because the reward function giving a 0 reward during the episode and a +1 reward only at the end of an episode (when the robot escapes from the maze), is not helping the agent (robot) to understand the environment (maze) better.

We want the agent to learn to escape from the maze as quickly as possible but due to lack of effective communication, the agent will not consider how long it takes to escape because the agent has not gained sufficient knowledge about the maze to decide what is right and what is wrong and thus, will keep taking random actions.

To effectively communicate what we need, we should give a negative reward for each step that the robot runs so that it should try to escape the maze as fast as it can, as the states that the robot visits a lot will get worse and worse values so it will want to move away from there and eventually find the goal.

Enroll
Page No:
Date: / /

Q8.)

Ans - Following are the 6 operators:

Op (ACTION: Rightshoe, PRECOND: RightSock ON,
EFFECT: Rightshoe On)

Op (ACTION: RightSock, EFFECT: RightSockOn)

Op (ACTION: Leftshoe, PRECOND: LeftsockOn,
EFFECT: Leftshoe On).

Op (ACTION: Leftshoe, EFFECT: LeftSock On).

Op (ACTION: Hat, EFFECT: Hat On)

Op (ACTION: Coat, EFFECT: Coat On).

Initial state has no literals.

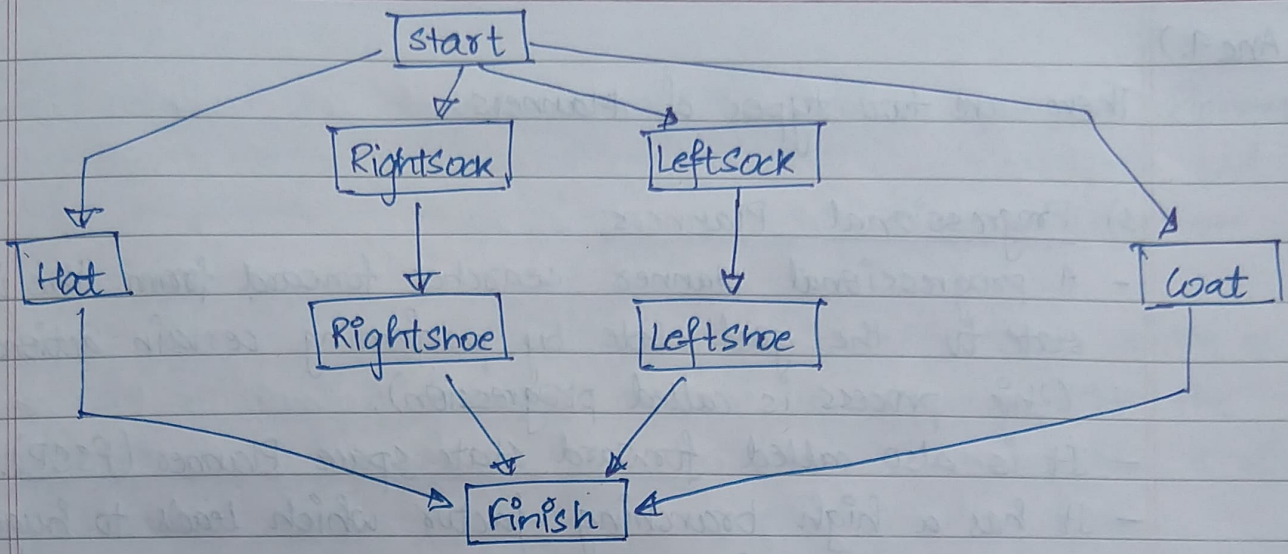
Goal state: RightshoeOn ∧ LeftshoeOn ∧ HatOn ∧ CoatOn

Partial-Order Plan

→ Initial Plan

Plan (STEPS { S_1: Op(ACTION: Start),
S_2: Op(ACTION: Finish, PRECOND: RightshoeOn ∧
LeftshoeOn ∧
HatOn ∧ CoatOn.

},
ORDERINGS: {$S_1 < S_2$})



Partial-Order Plan

In the partial-order plan, we have six steps in total.
∴ There are $6! = 720$ possible combinations.
Out of those 720 plans, half of them will have the ordering of Leftshoe step before Leftsock step which will violate the ordering constraint (∵ LeftsockOn is a PRECOND for Leftshoe).
This reduces the valid plans to $\frac{720}{2} = 360$.

Again, half of these will have the violating ordering of Rightshoe and Rightshoe steps (same as above).
∴ The total no. of plans that does not violate any constraints are $\frac{360}{2} = 180$.

⇒ There are 180 different linearizations (total order plans) for the above partial order plan solution.

Enroll
Page No:
Date: / /

Ans 9.)

There are two types of planners:

1) Progressional Planners
- A progressional planner searches forward from the initial state to the goal state by peoforming certain actions (this process is called progression).
- It is also called forward state space Planner (FSSP).
- It has a high branching factor which leads to huge search space size.

2) Regression Planners
- A regressional planner searches backward from goal state to initial state by finding the previous action to be done to achieve pevious goal or sub-goal (this process is called regression).
- It is also called Backward state space Planner (BSSP).
- It has a small branching factor.

There are two types of operators:

1) Refinement Operator
- They take partial plans and add constraints to it.
- They eliminate some plans but never add new plans.

2) Modification Operator
- It includes all other operators other than refinement operators.
- Adding a step, instantiating a previously unbound variable, etc are some of the Modification Operators.

Enroll
Page No:
Date: / /

POP Planner or Partial-Order Planner is a Regression Planner because it starts with goals that need to be achieved and work backwards to find operators that will achieve them.

A POP stores a set of ~~tactual~~ causal links. A causal link, written as $S_i \xrightarrow{c} S_j$, serves to record the purposes of steps in the plan (here purpose of S_i is to achieve the precondition c of S_j). Causal link specifies which actions meet which preconditions of other actions, and thus help in detecting threats.

In partial-order planning, threats refer to the ordering of actions / steps that threaten to break the connected actions (as specified by causal links) or violate the ordering constraints.

POP resolves these threats in 2 ways:

1) Promotion: Ordering the possible threat after the connection it threatens.
2) Demotion: Ordering the possible threat before the connection it threatens.

If at any point, POP fails to resolve a threat, it backtracks to a previous choice point.