Internal Assessment Test 2 – Answer Key, 2025-2026

Subject Name: Intelligent Systems and Machine Learning Algorithms
Subject Code: BEC515A
Date: 2/12/25

1. How does the f(n)=g(n)+h(n) formula in A* search combine path cost and heuristic value?
   The most widely known form of best-first search is called **A∗ search** .It evaluates nodes by combining *g(n)*, the cost to reach the node, and *h(n)*, the cost to get from the node to the goal:

$$f(n) = g(n) + h(n) \,.$$

Since *g(n)* gives the path cost from the start node to node *n*, and *h(n)* is the estimated cost of the cheapest path from *n* to the goal, we have

f(n) = estimated cost of the cheapest solution through *n* .

If the heuristic function *h(n)* satisfies certain conditions, A∗ search is both complete and optimal. The first condition we require for optimality is that *h(n)* be an admissible heuristic.

- An admissible heuristic is one that never overestimates the cost to reach the goal.
- Straight-line distance in the tourist problem is admissible because the shortest path between any two points is a straight line, so the straight line cannot be an overestimate.

A second, slightly stronger condition called consistency (or sometimes monotonicity) is required only for applications of A∗ to graph search. A heuristic *h(n)* is consistent if, for every node *n* and every successor *n'* of *n* generated by any action *a*, the estimated cost of reaching the goal from *n* is no greater than the step cost of getting to *n'* plus the estimated cost of reaching the goal from *n'*.This is a form of the general triangle inequality, which stipulates that each side of a triangle cannot be longer than the sum of the other two sides. A∗ has the following properties:

- The tree-search version of A ∗ is optimal if h(n) is admissible, while the graph-search version is optimal if h(n) is consistent.

We show that the graph-search version is optimal

- if h(n) is consistent, then the values of f(n) along any path are nondecreasing.

From the two preceding observations, it follows that the sequence of nodes expanded by A∗ using GRAPH-SEARCH is in nondecreasing order of f(n). Hence, the first goal node selected for expansion must be an optimal solution because f is the true cost for goal nodes and all later goal nodes will be at least as expensive. The fact that f-costs are nondecreasing along any path also means that we can draw contours in the state space, just like the contours in a topographic map. Inside the contour labeled 400, all nodes have f(n) less than or equal to 400, and so on. If C∗ is the cost of the optimal solution path, then we can say the following: • A∗ expands all nodes with f(n).

For problems with constant step costs, the growth in run time as a function of the optimal solution depth d is analyzed in terms of the absolute error or the relative error of the heuristic.

The absolute error is defined as $\Delta \equiv h* - h$, Where h∗ is the actual cost of getting from the root to the goal, and the relative error is defined as $\equiv (h* - h)/h*$.

For constant step costs, we can write this as O(b d),whered is the solution depth. For almost all heuristics in practical use, the absolute error is at least proportional to the path cost h∗, so is constant or growing and the time complexity is exponential in d.

2. Discuss the issues in Find-S algorithm. Discuss the difference between Find S and Candidate Elimination Algorithm.

<u>Issues in Find-S algorithm</u>

Has the learner converged to the correct target concept?

Although FIND-S will find a hypothesis consistent with the training data, it has no way to determine whether it has found the ***only*** hypothesis in ***H*** consistent with the data (i.e., the correct target concept), or whether there are many other consistent hypotheses as well.

Why prefer the most specific hypothesis?

In case there are multiple hypotheses consistent with the training examples, FIND-S will find the most specific. It is unclear whether we should prefer this hypothesis over the most general or some other hypothesis of intermediate generality.

Are the training examples consistent?

In most practical learning problems there is some chance that the training examples will contain at least some errors or noise. Such inconsistent sets of training examples can severely mislead FIND-S, given the fact that it ignores negative examples. We would prefer an algorithm that could at least detect when the training data is inconsistent and, preferably, accommodate such errors.

What if there are several maximally specific consistent hypotheses?

In the hypothesis language H for the ***EnjoySport*** task, there is always a unique, most specific hypothesis consistent with any set of positive examples. However, for other hypothesis spaces there can be several maximally specific hypotheses consistent with the data.

Find-S algorithm tries to find a hypothesis that is consistent with positive instances, ignoring all negative instances. As long as the training dataset is consistent, the hypothesis found by this algorithm may be consistent.

The algorithm finds only one unique hypothesis, wherein there may be many other hypotheses that are consistent with the training dataset.

Find-S Algorithm	**Candidate Elimination Algorithm**
Finds **one most-specific** hypothesis consistent with positive examples.	Finds **all possible hypotheses** consistent with both positive and negative examples.
Starts with the **most specific** hypothesis.	Maintains **two boundaries**: S (specific boundary) and G (general boundary).
Uses only positive training examples.	Uses **both positive and negative** training examples.
Hypothesis becomes **more general** when a positive example does not fit.	Updates both **S (generalizes)** and **G (specializes)** depending on the example.
Does not consider negative examples.	Negative examples are used to **specialize G** and remove inconsistent hypotheses.
Produces a **single final hypothesis**.	Produces a **version space**, a set of all consistent hypotheses.
Simple, fast, easy to implement.	More complex, computationally heavier.
Not suitable for noisy data.	Sensitive to noise but can detect contradictions.
Gives an **incomplete** view of the hypothesis space.	Gives a **complete** view of the hypothesis space.

3. Explain hill climbing algorithm in AI with examples.

Hill climbing is sometimes called **greedy local search** because it grabs a good neighbor state without thinking ahead about where to go next. For example, from the state in Figure , it takes just five steps to reach the state in Figure, which has h=1 and is very nearly a solution. Unfortunately, hill climbing often gets stuck for the following reasons: **Local maxima**: a local maximum is a peak that is higher than each of its neighboring states but lower than the global maximum. Hill-climbing algorithms that reach the vicinity of a local maximum will be drawn upward toward the peak but will then be stuck with nowhere else to go. the state in Figure is a local maximum (i.e., a local minimum for the cost h); every move of a single queen makes the situation worse. **Ridges**: a ridge is shown in Figure . Ridges result in a sequence of local maxima that is very difficult for greedy algorithms to navigate. **Plateaux**: a plateau is a flat area of the state-space landscape. It can be a flat local maximum, from which no uphill exit exists, or a **shoulder**, from which progress is possible. A hill-climbing search might get lost on the plateau.

Stochastic hill climbing chooses at random from among the uphill moves; the probability of selection can vary with the steepness of the uphill move. This usually converges more slowly than steepest ascent, but in some state landscapes, it finds better solutions.

First-choice hill climbing implements stochastic hill climbing by generating successors randomly until one is generated that is better than the current state. This is a good strategy when a state has many (e.g., thousands) of successors.

The hill-climbing algorithms described so far are incomplete—they often fail to find a goal when one exists because they can get stuck on local maxima.

Random-restart hill climbing adopts the well-known adage, "If at first you don't succeed, try, try again." It con ducts a series of hill-climbing searches from randomly generated initial states,1 until a goal is found.

- In each of the previous cases (local maxima, plateaus & ridge), the algorithm reaches a point at which no progress is being made.
- A solution is to do a **random-restart hill-climbing** - where random initial states are generated, running each until it halts or makes no discernible progress. The best result is then chosen.

4. Define Machine Learning. Explain different types of Machine Learning problems. Discuss the issues involved in machine learning.

A sub-branch of AI. "Machine Learning is the field of study that gives the computers ability to learn without being explicitly programmed".

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

1. Supervised Learning

   Classification
   Regression
2. Unsupervised Learning

   Cluster Analysis
   Association mining
   Dimensionality reduction
3. Semi-supervised Learning
4. Reinforcement Learning

Supervised Learning algorithms use labelled data

The input attributes are called independent variables. The target attribute is called dependent variable. Supervised learning has two methods

1. Classification
   - The target attribute is a discrete variable (aka categorical, labels etc.)
   - Some classification algorithms are: Decision Tree, Random Forest, Support Vector machines (SVM), Artificial Neural Networks (ANN)
2. Regression

The target attribute is a continuous variable. In unsupervised learning data do not have target values or labels. Examples of unsupervised learning are cluster analysis and dimensionality reduction

- Cluster Analysis
  - Aims to group objects into disjoint clusters or groups
  - Two key clustering algorithms are:
    - K-means algorithm
    - Hierarchical algorithm
- Dimensionality reduction
  - Converts high dimensional data into lower dimensional data
- Semi-supervised learning
  - Dataset has a large set of unlabelled data and some labelled data
- Reinforcement Learning
- Reinforcement Learning allows the agent to interact with environment to get rewards.

ISSUES in Machine Learning:

The quality of the learning systems depends on the quality of data. Some of the challenges with this are:

- Problems: Problems have to be well posed.
- Availability of large set of data
- High computational power required
- Complexity of algorithms
- Bias/variance

5. Write Find -S algorithm. Apply the Find-S for table Q1(c) to find maximally specific hypothesis.

Eyes	Nose	Head	Fcolor	Hair?	Smile?(TC)
Round	Triangle	Round	Purple	Yes	Yes
Square	Square	Square	Green	Yes	No
Square	Triangle	Round	Yellow	Yes	Yes
Round	Triangle	Round	Green	No	No
Square	Square	Round	Yellow	Yes	Yes

Table Q1(c)

5. Find S algorithm.

i) The hypothesis space is first initialized $h_0 = \{\phi, \phi, \phi, \phi \ldots\}$ depending on the number of input attributes present.

ii) The first input is also initialized which in turn becomes the next heuristic.

iii) The second positive instance is input and compared to the obtained heuristic. defined as next

iv) The values that don't match are replaced by an accept case.

v) This process is repeated until all positive instances have been utilized giving rise to the final heuristic.

vi) Find S algorithm does not work for negative instances

Table (1C)

$h_0 = \{\phi, \phi, \phi, \phi, \phi\}$

$x_1 = \{$ Round, Triangle, Round, Purple, Yes $\}$

$h_1 = \{$ Round, Triangle, Round, Purple, Yes $\}$

$x_2 = \{$ Square, Triangle, Round, Yellow, Yes $\}$

$h_2 = \{$?, Triangle, Round, ?, Yes $\}$

6. Outline the concepts involved in i) Measuring accuracy using Cross-Validation ii) Confusion Matrix iii) Precision and Recall.

Measuring Accuracy using Cross-Validation

Cross-validation is a technique used to evaluate the generalization performance of a machine learning model. It addresses the limitations of a simple train-test split by providing a more robust estimate of how the model will perform on unseen data.

- **Concept:**
  The dataset is divided into multiple "folds" (e.g., k-folds). The model is trained and tested iteratively, where in each iteration, a different fold serves as the test set, and the remaining folds are used for training. The performance metrics (including accuracy) from each iteration are then averaged to provide a more reliable and less biased estimate of the model's performance.
- **Purpose:**
  To mitigate overfitting and underfitting by ensuring the model's performance is not overly dependent on a specific train-test split, leading to a more robust evaluation of its ability to generalize.

ii) Confusion Matrix

A confusion matrix is a table that summarizes the performance of a classification model on a set of test data where the true values are known. It allows for a detailed analysis of the types of errors made by the classifier.

- **Components:**
  - **True Positives (TP):** Correctly predicted positive instances.
  - **True Negatives (TN):** Correctly predicted negative instances.
  - **False Positives (FP):** Incorrectly predicted positive instances (Type I error).
  - **False Negatives (FN):** Incorrectly predicted negative instances (Type II error).
- **Purpose:**
  To visualize and analyze the performance of a classification model, providing insights into which classes are being confused and the specific types of errors the model is making.

iii) Precision and Recall

Precision and Recall are key metrics derived from the confusion matrix, offering a more nuanced understanding of a model's performance than simple accuracy, especially in imbalanced datasets.

- **Precision:** Measures the proportion of correctly predicted positive instances out of all instances predicted as positive.
  Code
  ```
  Precision = TP / (TP + FP)
  ```
- **Purpose:**
  To assess the "exactness" of the model, answering the question: "Of all the instances predicted as positive, how many were actually positive?" High precision indicates a low rate of false positives.
- **Recall (Sensitivity):**
  Measures the proportion of correctly predicted positive instances out of all actual positive instances.
  Code
  ```
  Recall = TP / (TP + FN)
  ```

- **Purpose:** To assess the "completeness" of the model, answering the question: "Of all the actual positive instances, how many were correctly identified?" High recall indicates a low rate of false negatives.

7. Define the following terms with reference to machine learning (i) Concept Learning (ii) Inductive learning hypothesis (iii) Version Space (iv) General Boundary (vi) Specific boundary.

Concept learning is defined as: "Given a set of hypothesis, the learner searches through the hypothesis space to identify the best hypothesis that matches the target concept"
Concept learning requires three things

1. Input: Training dataset
2. Output: Target concept or target function
3. Test instances

Concept learning in machine learning refers to the process of teaching a machine to identify and recognize patterns from specific examples or data points. In simple terms, concept learning involves learning a general rule from a set of observed instances. For example, if you show a machine many pictures of cats, it will learn to recognize the concept of a "cat" and apply that knowledge to identify new cat pictures. Concept learning helps machines generalize from data. Instead of memorizing each example, it creates a broader understanding that can be applied to unseen situations.

The inductive learning hypothesis.
Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples.

The version space,
It is denoted as VSHVD, with respect to hypothesis space H and training examples D, is the subset of hypotheses from H consistent with the training examples in D.

VSH = {h E H | Consistent (h, D)]

The general boundary G, with respect to hypothesis space H and training data D, is the set of maximally general members of H consistent with D.

$$G \equiv \{g \in H | Consistent(g, D) \wedge (\neg\exists g' \in H)[(g' >_g g) \wedge Consistent(g', D)]\}$$

The specific boundary S, with respect to hypothesis space H and training data D, is the set of minimally general (i.e., maximally specific) members of H consistent with D.

$$S \equiv \{s \in H | Consistent(s, D) \wedge (\neg\exists s' \in H)[(s >_g s') \wedge Consistent(s', D)]\}$$

Version Spaces

A hypothesis h is **consistent** with a set of training examples D of target concept c if and only if $h(x) = c(x)$ for each training example $\langle x, c(x) \rangle$ in D.

$$Consistent(h, D) \equiv (\forall \langle x, c(x) \rangle \in D) \; h(x) = c(x)$$

The **version space**, $VS_{H,D}$, with respect to hypothesis space H and training examples D, is the subset of hypotheses from H consistent with all training examples in D.

$$VS_{H,D} \equiv \{h \in H | Consistent(h, D)\}$$

Question Paper:

USN										

Internal Assessment Test 2 – Nov. 2025

Sub:	Intelligent Systems and Machine Learning Algorithms							Code:	BEC515A
Date:	02/ 12 / 2025	Duration:	90 mins	Max Marks:	50	Sem:	V	Branch:	ECE

	Answer any 5 full questions	Marks	CO	RBT
1	How does the f(n)=g(n)+h(n) formula in A* search combine path cost and heuristic value?	[10]	CO3	L1
2	Discuss the issues in Find-S algorithm. Discuss the difference between Find S and Candidate Elimination Algorithm.	[10]	CO3	L1
3	Explain hill climbing algorithm in AI with examples.	[10]	CO3	L2

USN										

Internal Assessment Test 2 – Nov. 2025

Sub:	Intelligent Systems and Machine Learning Algorithms							Code:	BEC515A
Date:	02/ 12 / 2025	Duration:	90 mins	Max Marks:	50	Sem:	V	Branch:	ECE

	Answer any 5 full questions	Marks	CO	RBT
1	How does the f(n)=g(n)+h(n) formula in A* search combine path cost and heuristic value?	[10]	CO3	L1
2	Discuss the issues in Find-S algorithm. Discuss the difference between Find S and Candidate Elimination Algorithm.	[10]	CO3	L1
3	Explain hill climbing algorithm in AI with examples.	[10]	CO3	L2