Homework 1

Manyara Bonface Baraka - mbaraka

February 4, 2025

Part I: Theoretical Questions (40 points – 1 point for each question unless stated otherwise)

1. For each of the following assertions, say whether it is true or false and support your answer with examples or counterexamples where appropriate.

   (a) An agent that senses only partial information about the state cannot be perfectly rational.

   False: The agent can be perfectly rational as long as it makes the best decision based on its current knowledge.

   Example: A self-driving car can decide to slow down at the corner as much as it doesn't see what's around the corner.

   (b) There exist task environments in which no pure reflex agent can behave rationally.

   True Some task next decision rely on history and hence pure reflex model can't handle.

   Example Maze box problem, a pure reflex agent will stay in a loop if it doesn't track its path before to find and exit.

   (c) There exists a task environment in which every agent is rational.

   False playing chess will have a winner and losing strategy this indicates some agents will act irrational compared to others.

   (d) The input to an agent program is the same as the input to the agent function.

   False Input agent function is the entire history of the percepts while inpute to an agent program is the algorithm.

   (e) Every agent function is implementable by some program/machine combination.

   False Some agent function are uncomputable due to infinite state spaces.

(f) Suppose an agent selects its action uniformly at random from the set of possible actions. There exists a deterministic task environment in which this agent is rational.

True If the task environment rewards random exploration, then the rando-action agent can be rational.

(g) It is possible for a given agent to be perfectly rational in two distinct task environments.

True if and only if the agent maximised each expected utility in making decisions.

Example sorting aligorithm will work well in both numerical and alphabetic

(h) Every agent is rational in an unobservable environment.

False There is no information to act rationally in an unobservable environment.

(i) A perfectly rational poker-playing agent never loses.

False poker maximized the expected winnings however poker has a lot of randomness and hidden information that can lead to a perfect agent loss.

2. For each of the following activities, give a PEAS description of the task environment and characterize its properties.

(a) Playing soccer.

- **PEAS**
  - **Performance**: Goal score, defence
  - **Environment** Soccer field, ball, players
  - **Actuators** Legs, arms for goalkeepers, communication with teammates
  - **Sensors** Vision, touch, hearing
- **Characterization**
  - **Partially observable** Players don't have full information about the opponent's movements.
  - **Multiagent** opponent and teammates have independent strategies
  - **Sequential** Past actions affect future performance
  - **Stochastic** ball movement and opponent actions are random
  - **Continous** Positions, ball trajectories, and movements are continuous.

(b) Exploring the subsurface oceans of Titan.

- **PEAS**
  - **Performance** data collection accuracy, life detection
  - **Environment** temperature, ice-covered ocean

- **Actuators** sensors, robot arms
- **Sensors** camera, pressure sensors

- **Characterization**
  - **Partial Observable** unknown underwater obstacles
  - **Stochastic** ocean currents are random
  - **Sequential** actions like a movement info future choices.

(c) Shopping for used AI books on the Internet.

- **PEAS**
  - **Performance** Price, delivery time
  - **Environment** Online shop, product lists
  - **Actuators** Keyboard, mouse, payment
  - **Sensors** Web page, book details
- **Characterization**
  - **Partially Observable**, not all book or seller reliability are visible
  - **Stochastic** prices and availability are unpredictable
  - **Sequential** past purchase and searches influence future recommendations
  - **Dynamic** the prices and listings change

(d) Playing a tennis match.

- **PEAS**
  - **Performance** Points won, match outcome
  - **Environment** Tennis court, players
  - **Actuators** racket movements, arms, legs
  - **Sensors** Vision, hearing
- **Characterization**
  - **Partialy observable** Opponenent decision and swing is not fully known
  - **Sequential** Each action affect next decision
  - **Dynamic** the game state changes continuously
  - **Multiagent** interaction with an opponent
  - **Continous** movements and strokes are continuous

(e) Practicing tennis against a wall.

- **PEAS**
  - **Performance** Ball control
  - **Environment** Wall, ball, court
  - **Actuators** Arms, leg, racket
  - **Sensors** Vision, hearing
- **Characterization**

  - **Fully Observable** The ball trajectory and wall position are known
  - **Single agent** no opponent
  - **Sequential** past strokes affect future rallies
  - **Dynamice** ball moves even if the player does nothing.

(f) Performing a high jump.

- **PEAS**
  - **Performance** landing control, height cleared
  - **Environment** Track, body mechanics
  - **Actuators** Legs, arms, body movement
  - **Sensors** Vision, touch
- **Characterization**
  - **Fully Observable** bar height and jump trajectory are known
  - **Sequential** preparation affect the jump
  - **Single agent** no interaction with others
  - **Static** environment doesn't change
  - **Continous** Motion, body mechanics and trajectory

(g) Knitting a sweater.

- **PEAS**
  - **Performance** Accuracy of stitch, symmetry, completion time
  - **Environment** Yarn, knitting tools, hand, parttern
  - **Actuators** hand, fingers, knitting needle
  - **Sensors** Touch , vision
- **Characterization**
  - **Fully observable** Stich and yarn are visible
  - **Single agent** no other agent
  - **Sequantial** next stitch depends on previous
  - **Static** unless manipulated the yarn remains constant
  - **Deterministic** same action produce same stiche

(h) Bidding on an item at an auction.

- **PEAS**
  - **Performance** Winning the item at low price
  - **Environment** Other bidders, auctioneer, auction house
  - **Actuators** Raising hand(physical auction) or click (online)
  - **Sensors** Seeing, hearing, price updates
- **Characterization**
  - **Partially observable** other bidders budget and strategies are unknown
  - **Multiagent** other bidders

- **Stochastic** Other bidders action are unpredictable
- **Sequantial** Each bid affect future bid
- **Dynamic** Auction progresses even if the agent does nothing

3. This question explores the differences between agent functions and agent programs.

   (a) Can there be more than one agent program that implements a given agent function? Give an example, or show why one is not possible.

   Yes different program can produce same agent function using different algorithms

   Example Vaccum cleaner function can have different programs like

   - use neural network trained on past experience
   - use rule-based system with conditional statements
   - An explicit loopup table mapping percepts to actions

   (b) Are there agent functions that cannot be implemented by any agent program?

   Yes these happens on functions due to infinite percept space, uncomputability and real-time constraints.

   (c) Given a fixed machine architecture, does each agent program implement exactly one agent function?

   No a single agent program can implement multiply agent function depending with inputs or internal state.

   (d) Given an architecture with n bits of storage, how many different possible agent programs are there?

   Since each bit is either 0 or 1, then there are n bits, the total number will be

   $$2^n$$

   (e) Suppose we keep the agent program fixed but speed up the machine by a factor of two. Does that change the agent function?

   No Speeding up does not change the function but reduces the computational time but the function remains unchanged.

4. Implement a performance-measuring environment simulator for the vacuum-cleaner world depicted in the following figure. Your implementation should be modular so that the sensors, actuators, and environment characteristics (size, shape, dirt placement, etc.) can be changed easily. (6 points)

```
To implement the vacuum cleaner:
1. Define the environment
    - Locations A and B with either random or predefined dirt placement
    - The agent will be placed at a random location
```

```
2. Define the performance metric
    - Action count = 0 --> Tracking the number of moves and cleaning actions
    - Dirt_cleaned = 0 // Tracking the numberof cleaned places
    - Total dirt = counting the total dirty locations at the the start

3. While loop when the environment is not fully clean:
    a. Sense the current location
        - If dirty -- clean the location
            - increase the Dirt_cleand count
            - increase the Action_count
        - Else -- move to another location
            - increase the Action_count

4. Compute the performance score
    - Efficiency =  Dirt_cleaned / Total_dirt
    - Action Efficiency = Total_dirt / Action_count

5. Output the performance metrics
    - Print the total actions taken
    - print the number of dirt patches
    - print the efficiency score
```

5. Consider a modified version of the vacuum environment in Question 4, in which the geography of the environment—its extent, boundaries, and obstacles—is unknown, as is the initial dirt configuration. (The agent can go Up and Down as well as Left and Right) (12 points – 3 points each)

   (a) Can a simple reflex agent be perfectly rational for this environment? Explain.

   No Since the modified environment is partially observable with obstacles, the simple reflex agent cannot be perfectly rational in this environment since it doesn't keep memory of the past state. If it encounters an obstacle it has no memory of where it has been and maybe stack in loops. Simple reflex agent cannont be rational since rationallity requires maximum performance in which it needs memory and reasoning.

   (b) Can a simple reflex agent with a randomized agent function outperform a simple reflex agent? Design such an agent and measure its performance on several environments.

   Yes and No - in some cases and not all cases: Randomisez agent has random movements this reduces the chances of getting stuck however, randomness does not guarantee effeciency,the agent might still revisit clean areas or move inefficiently.

```
1. Sense current location:
    - If dirty -> clean
    - Else -> move randomly (Left, Right, Up or Down)
2. If the next move hits an obstacle or boundary:
    - Choose another random move
3. Continure until all dirt is cleaned
4. Evaluate performance metrics as above
```

(c) Can you design an environment in which your randomized agent will perform poorly? Show your results.

Yes an environment like a narrow corridor or maze-like environment will cause the random agent to have a poor performance since, it might be move inefficiently by revisiting the same space more frequently additionally it will waste more moves hitting the wall and obstacles.

The expected performance:

- High number of moves before all dirt is cleaned
- Random motion increases inefficiency
- Wasted actions in hitting walls

(d) Can a reflex agent with state outperform a simple reflex agent? Design such an agent and measure its performance on several environments. Can you design a rational agent of this type?

Yes this is because the reflex agent with state remembers the visited locations and minimizes unnecessary movements.

Designin the reflex agent with state

```
1. Sense the current location:
    - If dirty -> Clean and mark as cleaned
    - Else -> Choose a move that minimizes revisiting cleaned areas
2. Keep memory of the visited locations:
    - Store previous positions
    - Don't move to the already cleaned areas if possible
3. If an obstacle is encountered
    - Store its location in memory
    - Choose an alternative path
4. Stop when all reachable areas are clean
```

Rational agent of this type Yes, this is because a rational agent should map the environment dynamically, use efficient search algorithm to cover all dirty areas with minimal moves and plan the paths with reasoning instead of moving randomly.

Part II: Python Problems

Utility-Based Personalized Recommendation Agent

The key factors considered for utility calculation are:

1. **Category Match (40%)** – If a product belongs to a category the user prefers, it receives a **higher score**.
2. **User Ratings Influence (50%)** – If a product has been rated before, its previous rating affects its utility score. For new products, an average rating is used.
3. **Price Penalty (10%)** – Higher-priced products are penalized slightly to encourage affordability without ignoring quality.

The final **utility score** is computed using a weighted sum of these factors:

$$Utility = 0.4 \times \text{CategoryScore} + 0.5 \times \text{RatingScore} - 0.1 \times \text{PricePenalty} \quad (1)$$

Why These Factors and Weights?

Category Preference (40%)

- Users are **more likely to purchase** products that align with their interests.
- The weight (40%) ensures category preference influences recommendations but doesn't dominate other factors.

User Ratings (50%)

- A **highly rated product** should be prioritized.
- This factor is **most influential (50%)** because past user satisfaction directly reflects product desirability.

Price Penalty (10%)

- Users might avoid **overly expensive** products.
- A **lower weight (10%)** ensures affordability is considered but **does not overshadow** product quality.