Artificial Intelligence Lab Manual

Group Name: F A N U M
Gaganpreet Singh[202211023], Pragya Pranati[202211068], Suryansh Singh Raghuvansh[202211093]

***Abstract*—This lab report presents four experiments exploring various AI techniques in problem-solving scenarios. The first experiment, Rabbit Leap Problem, investigates whether a group of rabbits can cross each other on a set of stones without stepping into the water, using a forward-only movement and leap strategy. The second experiment, Plagiarism Checker, utilizes the A* search algorithm to detect plagiarism by aligning text sequences between two documents. The third experiment, K-SAT Problem, focuses on the generation and solving of uniform random k-SAT problems, comparing the performance of Hill-Climbing, Beam Search, and Variable Neighborhood Descent algorithms. Finally, the fourth experiment, Puzzle Solving, aims to create an AI agent capable of solving a complex puzzle by applying state space search methods. Through these experiments, this report demonstrates the effectiveness of AI techniques in solving both combinatorial and real-world problems.**

Keywords: AI algorithms, Beam Search, Variable Neighborhood Descent, state space search, puzzle solving, heuristic functions

I. Introduction

In Artificial Intelligence (AI) plays a crucial role in solving complex problems across various domains, from natural language processing to combinatorial optimization. This lab focuses on applying AI algorithms to four different types of problems, each requiring a unique approach for effective problem-solving. In Experiment 1, the challenge of orchestrating movement patterns for rabbits demonstrates AI's role in pathfinding and optimization. Experiment 2 addresses plagiarism detection, a critical issue in academic and professional environments, using heuristic search for efficient alignment of text sequences. Experiment 3 delves into solving k-SAT problems, a foundational problem in computational theory, by employing heuristic-based algorithms to explore solution spaces. Lastly, Experiment 4 explores puzzle-solving through AI agents, where the goal is to construct a search-based solution to navigate the puzzle state space efficiently. Collectively, these experiments highlight the diverse applications of AI and showcase how intelligent algorithms can be leveraged for both theoretical and practical problem-solving.

Experiment 1: Matchbox Educable Naughts and Crosses Engine

Read the reference on MENACE by Michie and check for its implementations. Pick the one that you like the most and go through the code carefully. Highlight the parts that you feel are crucial. If possible, try to code the MENACE in any programming language of your liking.

Experiment 2: Reinforcement Learning [The Binary Bandit Problem]

- Consider a binary bandit with two rewards 1-success, 0-failure. The bandit returns 1 or 0 for the action that you select, i.e. 1 or 2. The rewards are stochastic (but stationary). Use an epsilon-greedy algorithm discussed in class and decide upon the action to take for maximizing the expected reward. There are two binary bandits named binaryBanditA.m and binaryBanditB.m are waiting for you.
- Develop a 10-armed bandit in which all ten mean-rewards start out equal and then take independent random walks (by adding a normally distributed increment with mean zero and standard deviation 0.01 to all mean-rewards on each time step). function [value] = bandit-nonstat(action)
- The 10-armed bandit that you developed (bandit-nonstat) is difficult to crack with a standard epsilon-greedy algorithm since the rewards are non-stationary. We did discuss how to track non-stationary rewards in class. Write a modified epsilon-greedy agent and show whether it is able to latch onto correct actions or not. (Try at least 10000 time steps before commenting on results)

II. Experiment 1: Matchbox Educable Naughts and Crosses Engine

A. Problem Statement Description

The MENACE (Matchbox Educable Noughts And Crosses Engine) by Donald Michie is an early example of machine learning applied to game-playing. MENACE learns to play Noughts and Crosses (Tic-tac-toe) through trial-and-error, using matchboxes filled with beads representing possible game moves. Here's a breakdown of the MENACE system and its implementation principles:

- **Representation of Game States:** Each matchbox represents a unique board configuration for the player. There are 287 distinct boxes accounting for the various configurations when MENACE plays first.
- **Beads for Moves:** Different colored beads in the matchboxes represent potential moves. Each color corresponds to an empty position on the board.
- **Selection Process:** To make a move, the matchbox corresponding to the current game state is shaken, and a bead is randomly selected, determining the move.
- **Reinforcement Learning:** After the game concludes:
  - If MENACE wins, additional beads of the chosen colors are added to the boxes used during the game, making successful moves more likely in the future.
  - If MENACE loses, beads representing the chosen moves are removed, reducing the likelihood of repeating unsuccessful strategies.

B. Solutions

In exploring the MENACE (Matchbox Educable Noughts and Crosses Engine) developed by Donald Michie, I focused on the implementation that uses a simple and intuitive approach to mimic learning in games. The core concept involves representing the game board states and corresponding actions within matchboxes, each associated with a specific board configuration. I carefully examined the code, identifying crucial components such as the structure of the matchboxes, the mechanism for selecting moves based on the frequency of past outcomes, and the update rules that modify the probabilities of moves after each game.

For the implementation, the code consists of modular functions: one for representing the game state, another for the matchbox logic, and additional functions for training against either a random opponent or a human player. The matchbox structure was pivotal in enabling the learning process, as it dynamically adjusts the likelihood of successful moves based on previous interactions. I ensured that the code handles input gracefully, allowing for interactive play while maintaining an effective training mechanism. Overall, my design focused on maintaining clarity and simplicity while capturing the essence of MENACE's learning capabilities.

III. Experiment 2: Reinforcement Learning [The Binary Bandit Problem]

A. Problem Statements

The three case problems of the Reinforcement Learning Binary Bandit Problem explore different strategies for optimizing decision-making in uncertain environments. These cases are:

- **Case 1:** It involves a basic binary bandit setup where an epsilon-greedy strategy is employed, allowing the agent to balance exploration and exploitation to identify the arm with the highest mean reward.
- **Case 2:** It extends this by implementing a 10-armed bandit where the mean rewards start equal and undergo independent random walks, presenting a challenge for standard algorithms due to the non-stationary nature of rewards.
- **Case 3:** It introduces a modified epsilon-greedy agent capable of adapting to non-stationary environments by incorporating epsilon decay, thus improving its ability to track changing rewards and ultimately determining optimal actions over a longer series of interactions.

Collectively, these case problems illustrate the complexities of reinforcement learning in dynamic settings and the importance of adaptive strategies for effective decision-making.

B. Solutions

The approach to the three problem cases of the Binary Bandit Problem involves implementing reinforcement learning strategies to optimize decision-making in uncertain environments.

- **Case 1:** A basic epsilon-greedy algorithm was used to balance exploration and exploitation with a binary bandit, allowing the agent to gradually learn which of the two arms yields higher rewards. This was implemented by maintaining estimates of the mean rewards and updating them based on received feedback.
- **Case 2:** It expanded the scenario to a 10-armed bandit with non-stationary rewards, where all arms underwent independent random walks, making it challenging for traditional algorithms. Here, a separate 'NonStationaryBandit' class was implemented to simulate the reward structure, and the agent selected actions randomly to illustrate the difficulty in tracking true values.
- **Case 3:** It is an enhanced epsilon-greedy agent was developed, incorporating mechanisms to adapt to the changing nature of rewards, such as decreasing epsilon over time, thus improving its performance in identifying optimal arms.

The code for each case utilized classes to encapsulate the bandit logic, and various plotting functions were added to visualize the results, including the evolution of rewards, action counts, and true values for better analysis of the agents' performance.

C. Comparison

Every case corresponding code has generated graphs for easier comparison of performance and accuracy details

IV. References

1) Matchbox Educable Naughts and Crosses Engine [Link].
2) Reinforcement Learning: an introduction by R Sutton and A Barto (Second Edition) (Chapter 1-2) [Link].
3) Artificial Intelligence A Modern Approach By Stuart Russell , Peter Norvig[Link].
4) A First Course in Artificial Intelligence, Khemani D., Tata McGraw Hill, 2014.

V. GitHub Repository

The GitHub Repository to the the CS367 course lab(s) submissions can be accessed here