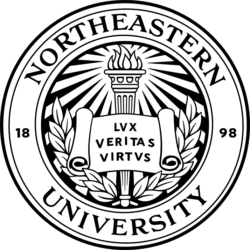
The Menace Project Report



Spring 2022

COURSE

Program Structures and Algorithms (INFO6205)

INSTRUCTOR

Prof. Robin Hillyard

PRESENTED BY

Ankur Thakur

(NUID: 002927572)

Akanksha Pawar

(NUID 002928432)

Himanshu Walia

(NUID 002960393)

Neha Chandel

(NUID 001593822)

TABLE OF CONTENTS

***INTRODUCTION ...................................................................................................................4***

* ***Aim .………………………...........................................................................................5***
* ***Approach …………………………………………………………………………………………………………6***

***PROGRAM ..........................................................................................................................6***

* **Data Structures & Classes ..............................................................................................6**
* **Algorithm …………………………………………………………………………………………………………………….7**
* **Invariants …………………………………………………………………………………………………………………….7**

***Flow Charts .........................................................................................................................7***

***OBSERVATIONS AND GRAPHICAL ANALYSIS ….....................................................................8***

***RESULTS AND MATHEMATICAL ANALYSIS ...........................................................................9***

***TEST CASES ........................................................................................................................11***

***CONCLUSION .....................................................................................................................13***

***REFERENCE.........................................................................................................................13***

INTRODUCTION

A simple board game Tic Tac Toe where human wisdom has shined for a long time for using human intelligence to implement machine intelligence. So many machine learning techniques have been explored, argued, developed, and exploited as researchers are extensively trying to work on these canonical problems.

The ideas were designed mechanically before the implementation of sophisticated learning models. Michie’s design, called MENACE, was a large pile of matchboxes that contained several beads and learned to play tic-tac-toe. Each matchbox represents a specific board layout of Tic Tac Toe. The menace is randomly optimized at the beginning, but after going through a few iterations it learns itself to favor the moves which are more successful in each situation. To facilitate this learning reinforcement is applied in which the machine is observed for its move and gets punished and rewarded. If the machine has done bad, the punishment is subjected to confiscation of the selected bead from each of the cells used during the play. By doing this it becomes less probable that the unsuccessful moves are repeated when any of these positions recur in future play.

The first move can be made by the Menace or human player when one randomly picks a bead out of the box representing the game’s current state. The human chooses random beads. The beads are adjusted when there is a failure or success. At the end of each game, if the menace loses each bead the menace used is removed from each box. If menace wins, beads of the same color are added to their respective box.

AIM OF THE PROJECT

* Implement “The Menace” by replacing matchboxes with values in a hash table (key will be the state of the game).
* Train the Menace by running games played against “human” strategy,   
  which is based upon the optimal strategy of the tic-tac-toe game.
* Values used in the project:
* Alpha (the number of beads in each matchbox at the start of the game—may be different for each move: first move, second move, etc.)
* Beta (the number of beads to add to the matchbox in the event of a win)
* Gamma (the number of beads to take to the matchbox in the event of a loss)
* Delta (the number of beads to add to the matchbox in the event of a draw)
* Human strategy: chooses optimal strategy with probability p\*. In the “zone”, chooses a random move.
* Implement logging:
* Log each training run with date/time, win/loss/draw, and p.
* Log every move taken by the Menace and its opponent for the final match.
* Uses SLF4j as the logging framework.

APPROACH

The key in the hash table is determined using the board state after each move and depending on that next move is chosen which is determined by training the system using a Reinforcement learning algorithm.

The correct move galvanizes the algorithm. Q-learning is also used to determine the rewards for each stem performed by the algorithm. We used a trial-and-error approach to train the agent because it does not use the transition probability distribution.

PROGRAM

DATA STRUCTURE & CLASSES

DATA STRUCTURE

In this menace model below are the data structures used:

* To contain matchboxes, the HashMap structure is used for each move Menace makes.
* To record boxes and beads List data structure is used.
* To represent the board 2D array is used.
* To represent each matchbox, where the position is used as the key and the beads as the value, HashTable is used.

CLASSES

TrainBot/TicTacToe.java

Main()- Entry point

Play.java

computePlay(): returns the best move for the move played. Contains all the possible win/lose/draw conditions

TicTacToeAPI.java

Winner()- checks all the win conditions

Classes/TicTacToe.java

InsertConfig()- inserts the current config into the dictionary

StorePlay()- stores the character in the gameboard at the player’s position

Wins(): Checks if player or computer has won the game

ALGORITHM

Reinforcement learning is also known as a semi-supervised learning model is an area of Machine Learning. It works by taking suitable actions to get the maximum reward in a particular situation. It is used by different software and machines to find the best possible behavior or path it should take in a specific situation. Reinforcement algorithm differs from Supervised learning in the way that in reinforcement learning we have an agent who decides which task to perform whereas in supervised learning there is an answer key in the training data. In reinforcement learning, since there is no training dataset, it learns from its experience.

INVARIANTS

The invariant is TicTacToe class in the project because it contains the board state. The number of O’s and X’s are related on the board. The relationship between O’s and X’s will hold even if the number of X’s and O’s are same.

FLOW CHARTS

Diagram

Description automatically generated

OBSERVATIONS & GRAPHICAL REPRESENTATION

OBSERVATIONS

The main observation is 2 planes, 3x3 board.  
player 1, the first plane represents the placements of Xs, and the second plane shows the placement of Os. The cell once occupied by a mark cannot be used again. The next state can be completely determined by the current state and action executed. The game environment has a finite number of states, percepts, and actions which makes the game environment static and discrete. It is also observed that the environment is sequential since the current move could affect future decisions. The agent was trained by playing the game of tic-tac-toe using reinforcement learning. To generate gameplay data, an untrained agent played against another untrained agent by making random moves. It is a two-player game with an opponent as a human agent, so it is a multiagent platform. The gameplay data contained the board state, the moves that were made, and who wins the game. Thus, the tic tac toe platform is sequential, deterministic, fully observable, static, discrete, and multi-agent.

Tic Tac Toe players have a possibility to fill in each of the nine entries with one of the two possible values an ‘X’ and an ‘O’.

So, the whole board can be broken down into 9 states:

S1: P1 places O1: P1 has 9 choices to do this move.

S2: P2 places X1: P2 has 8 choices to do this move.

S3: P1 places O2: P1 has 7 choices to do this move.

S4: P2 places X2: P2 has 6 choices to do this move.

S5: P1 places O3: P1 has 5 choices to do this move.

S6: P2 places X3: P2 has 4 choices to do this move.

S7: P1 places O4: P1 has 3 choices to do this move.

S8: P2 places X4: P2 has 2 choices to do this move.

S9: P1 places O5: P1 has 1 choice to do this move.

Given the above analysis, there are 9! Unique ways to fill out the grid.

9! = 362,880.

GRAPHICAL REPRESENTATION

Chart, bar chart

Description automatically generated

RESULTS AND MATHEMATICAL REPRESENTATION

RESULTS

Tic Tac Toe players have a possibility to fill in each of the nine entries with one of the two possible values an ‘X’ and an ‘O’.

So, the whole board can be broken down into 9 states:

S1: P1 places O1: P1 has 9 choices to do this move.

S2: P2 places X1: P2 has 8 choices to do this move.

S3: P1 places O2: P1 has 7 choices to do this move.

S4: P2 places X2: P2 has 6 choices to do this move.

S5: P1 places O3: P1 has 5 choices to do this move.

S6: P2 places X3: P2 has 4 choices to do this move.

S7: P1 places O4: P1 has 3 choices to do this move.

S8: P2 places X4: P2 has 2 choices to do this move.

S9: P1 places O5: P1 has 1 choice to do this move.

Given the above analysis, there are 9! Unique ways to fill out the grid.

9! = 362,880.

First, we trained Menace 300 plays, and below are the statistics that were performed after training.

Table

Description automatically generated

Then, we trained Menace 700 plays, and below are the statistics that were performed after training.

Table

Description automatically generated

MATHEMATICAL REPRESENTATION

Implementing reinforcement learning involves an agent, a set of states S, a set of actions A.

After Δt steps into the future, the agent will decide the next step. The weight for this step is calculated as ƔΔt, where Ɣ  (the discount factor) is a number between 0 and 1. (0 <= Ɣ <= 1) and has the effect of valuing rewards received earlier higher than those received later (reflecting the value of a "good start"). Ɣ  may also be interpreted as the probability of success (or survival) at every step Δt.

The algorithm, therefore, has a function that calculates the quality of a state–action combination:

Q: S x A → R

Before learning begins, Q is initialized to a possibly arbitrary fixed value (chosen by the programmer). Then, at each time t, the agent selects an action observes a reward enters a new state (that may depend on both the previous state and the selected action), and Q is updated. The core of the algorithm is a Bellman equation as a simple value iteration update, using the weighted average of the old value and the new information.

where is the reward received when moving from the state to the state, and ɑ is the learning rate 0< ɑ <=1.

Text

Description automatically generated with medium confidence

Where rt is the reward received when moving from the state st to the state st+1, and ɑ is the learning rate 0< ɑ <=1.

TESTCASES

Below is the screen sprint of all test cases passing successfully.

Text

Description automatically generated

LOG

The SLF4J logging abstractions are used because they are simple, flexible and works with all available logging frameworks. The purpose of the logging is used in our project to trace the Menace in each state.

CONCLUSION

When the model wasn't trained, it was giving random moves, but after training the model for more than 40,000 it has started to give correct moves. The use of reinforcement learning maximizes performance.

REFERENCES

Wikimedia Foundation. (2021, October 9). *Matchbox Educable Noughts and crosses engine*. Wikipedia. Retrieved April 27, 2022, from <https://en.wikipedia.org/wiki/Matchbox_Educable_Noughts_and_Crosses_Engine>

Wikipedia Q learning.

https://en.wikipedia.org/wiki/Q-learning

You Tube :

<https://www.youtube.com/watch?v=TtisQ9yZ2zo&t=2210s>