Minesweeper - Techniques of artificial intelligence

PROJECT REPORT

Anthony Zhou, Hamza EL Miri & Julien Baudru

March 31, 2022

Contents

1 Introduction

Following its integration by standard on Windows 3.1 systems in 1992, The Minesweeper quickly became one of the most popular puzzle games of the century. The goal of this single-player game is simple, the player must discover as many tiles in the game as possible while trying to never click on a mine. The player has clues about the potential positions of the mines, indeed each discovered tile contains a number indicating how many mines are in its neighborhood.

In these pages, we will present the different points we needed to create an artificial intelligence capable of playing the game described above in the most successful way possible.

2 Models

In this section, the different solutions found to solve the given problem will be presented and compared, these solutions are a solver (an algorithmic logical approach), a neural network and genetic algorithms.

2.1 Solver

Before implementing any machine learning technique, we first designed a solver for the minesweeper game. The interest of proceeding in this way is to be able to compare the results obtained by the different models tested during this section with a more classical algorithmic solution.

The way we designed this solver is as follows: TODO

2.2 Neural network

In a first step we tried to solve the given problem using a neural network algorithm. To do this we used the well known *tensorflow* library, the general idea for which we use this library is the following: From the values of the tiles on the current game board (i.e. a matrix of numbers going from 0 to 8 for the revealed tiles and -8 for the unknown tiles) this neural network will have to return a new matrix with, for each position of the tiles on the board, the probability that the tiles contain a mine.

TODO : Explain architecture choosen

2.2.1 Training

We first started training this model with 16 x 16 size game boards (see 3a) and we tried multiple combinations of hyper-parameters for this configuration, including the number of generated games the model takes as input, the number of training episodes and the batch size. However, despite the different combinations of these parameters, the maximum accuracy achieved by our model was 18%. We therefore chose to reduce the size of the game board to 8 x 8 (see 3b), which allowed us to reach a level of accuracy close to 50% in a first step. Figure 1 shows the difference in accuracy achieved as a function of the size of the game board.

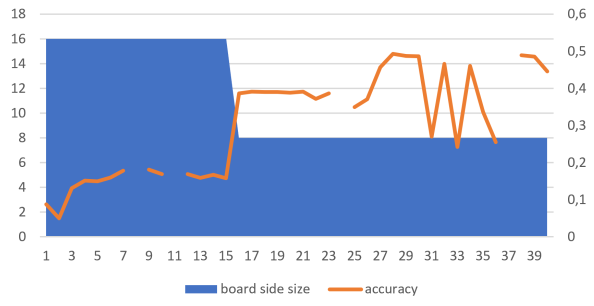


Figure 1: Accuracy by board size (16x16, 8x8)

There are several options for creating the set of inputs required by the neural network. The first one consists in generating randomly a large number of boards corresponding to the beginning of the game (option 1), the second one consists in creating sequences of game boards by making random choices (option 2) and the last one consists in creating sequences of game boards leading to a victory thanks to the solver (option 3). For each of these options, a matrix with positions of mines of each game board is also provided to the model.

Figure 2 shows the results obtained with 10000 sample game boards for the different options presented.

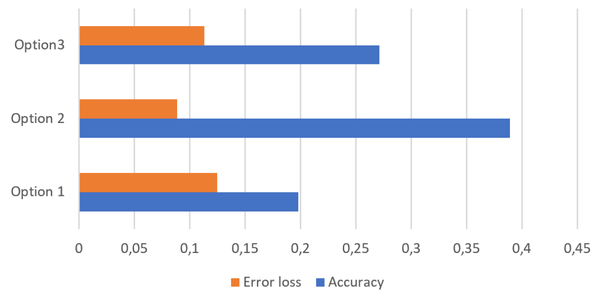


Figure 2: Accuracy and error loss by option

We notice that taking winning game sequences increases the accuracy of the model compared to a set of first game boards. However the best input in term of precision is the one constituted by suites of game generated by random moves (option 2).

2.2.2 Testing

2.3 Genetic algorithm

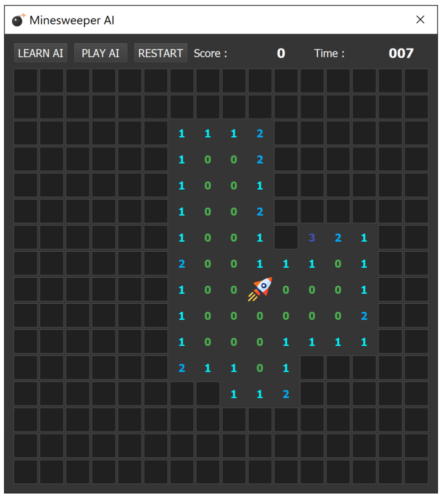
2.3.1 Training

2.3.2 Testing

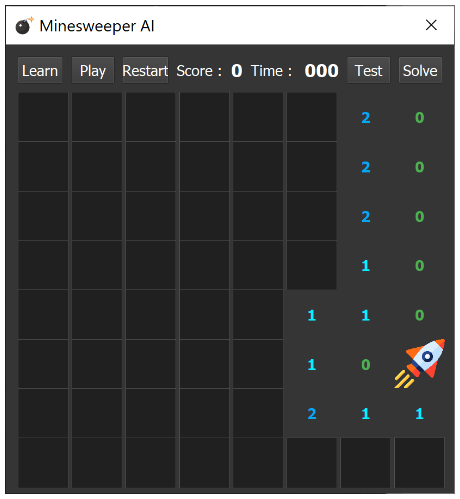
3 Comparisons

In this section we will compare and study the results obtained by the different models presented in the previous section.

4 Screenshots



(a) Screenshot board 16 x 16



(b) Screenshot board 8 x 8