GuessingGame

Manuel Hinz

May 21, 2022

Script and website

The game

GuessingGame takes place on a 8×8 grid world, where a single player has to guess the location of 4 rectangles (3×1, 5×1, 7×1, 2×2), which may be rotated (by 90 degrees) and are randomly placed into the world in a non-overlapping manner. The goal is to do so, while minimizing the number of guesses. Each guess reveals the state of a square (occupied by a square or empty). The game is won, if the player shot all squares which where occupied by rectangles.

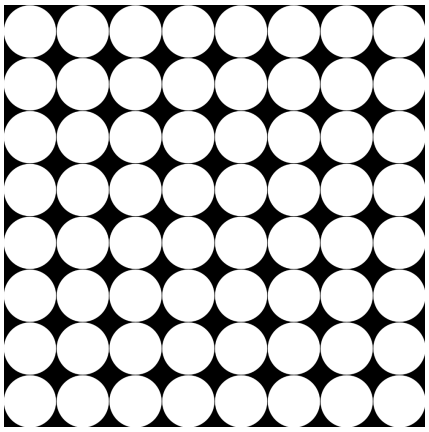

Figure 1: The 8×8 world

The approach

The agent, Myshkin, takes guesses s.t. he maximizes the expected information gain while he does not know the location and rotation of each rectangle and shoots the known, but not yet guessed, squares otherwise. Myshkin does so by considering all possible boards, which are consistent with the information already retrieved, and then calculating the entropy for each unknown square.

Interface

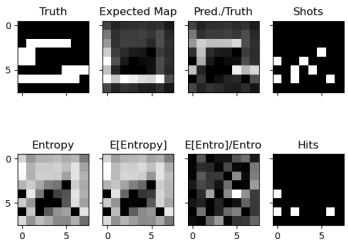

Figure 2: Myshkin interface

The interface saves 8 images after each move.

- Truth: The correct position of the rectangles.
- Expected map: The expected value of each square, where white indicates a higher expected value.
- Pred./Truth: The absolute difference of the truth and the expected map.
- Shots: The shots already made by Myshkin.
- Entropy: The entropy of each square.
- E[Entropy]: The entropy guessed by a subset of the possible boards.
- E[Entro]/Entro: The absolute difference of the entropy and the guessed entropy.
- Hits: The shots, which hit a square.

The expected entropy is not used in the algorithm (because we already the entropy gets calculated), but to evaluate the tradeoff between speed and performance when estimating the entropy.

Website

The script also saves a replay file, which can be used to view the guesses of Myshkin on this website. You can also play the game there yourself and see how your performance compares to his!

Code

The code can be found here.

Mathematical background

The entropy for a random variable with density ρ is defined as

$$\mathbb{H}(X) = -\mathbb{E}[\log_2 \rho].$$

It can be described as the expected amount of information retrieved by a random variable X in "bits".
Here our "random" variable is

$$X_{ij} \coloneqq \text{ a rectangle occupies the square at (i,j).}$$

Therefore we calculate the information we get by shooting at a square based on the previously known information. It follows that:

$$\mathbb{H}(X_{ij}) = -\left(\frac{n_i}{n} \cdot \log_2\left(\frac{n_i}{n}\right) + \frac{n_v}{n} \cdot \log_2\left(\frac{n_v}{n}\right)\right)$$

where n_i, n_v are the number of possible boards for which (i, j) is occupied by a square or not and $n \coloneqq n_i + n_v$.

Inspiration

At the height of the wordle hype, the following two mediums where especially influential to this project:

- Solving Wordle using information theory by Grant Sanderson
- This twitter thread by Tivadar Danka

Known limitations

- The data generation is quite slow and takes up quite a lot of memory.
- The first steps get calculated quite slow as well.
- The website displays the "won" alert before coloring the last square.