Mathematics In The Minesweeper

Rogger García

October 14, 2020

Abstract

In this paper, we explore different mathematical approaches in Minesweeper like computational algorithms, NP-Complexity, and matrix analysis. In the processes, I present strategies to satisfy computational complex algorithms such as heuristic h-neighbor adjacent.

Introduction

Go, Sudoku, some of the many games that have fascinated and challenged mathematicians and scientists. A game may be straightforward to play or a simple set of rules that a young boy could understand, but sometimes is the simplest things that have great things; emerging complex and intriguing problems, taking us to unexpected mathematical fields. When game researchers try to find techniques to understand how something works, sometimes it is needed to build mechanisms to get it. For example, AlphaGO is a computer video game that plays the board game Go . AlphaGO is capable of tackling problems similar to a human, analyzing patterns in humans, working with a combination of machine learning techniques and tree search algorithms. Chess. This game had fascinated computer science researchers, for how to successfully tackle each move human to formulate an Artificial Intelligent to win him. In the same sense, Deep Blue was a chess-playing computer developed by IBM that was capable of defeating the world chess champion, Gary Kasparov, in February of 1989. Minesweeper is another example of simple playing rules. In fact, Minesweeper is in a class of mathematical difficult problems known as NP-complete, these classes problems form part of Millennium Problems that may be earned by Clay Mathematics Institute with 1 million dollars if you demonstrate its solution.

In this paper, I will analyze different mathematical approaches designing algorithms and a matrix analysis that refers to Minesweeper. I will first provide a brief description of the game in Session One, in the Session Two Gauss Elimination Approach; Session Three NP-Complexity, and Chapter 4 Algorithm analysis .

1 Session One

1.1 Minesweeper Overview

Minesweeper is a simple-player puzzle video game developed by Robert Donner in 1989. The game consists in a n x m of grid-covered squares when the player must explore through a square revealing either a mine or an integer. This integer represents the number of mines adjacent to that clicked square. The game ends when the player probes a cell containing a mine, and the goal is to prove all uncovered squares that do not contain a mine.

Minesweeper allows the player to marl with a flag the position of the possible mine, but the game does not validate any flagged square, for this reason, higher levels of Minesweeper involve a greater degree of deductive reasoning: the number of mines increases.

Exists three levels in the game: beginning, intermediate, and expert. The beginning has a total of 10 mines and the board size is either 8 x 8, 9 x 9, or 10 x 10. Intermediate has 40 mines and also varies in size between 13 x 15 and 16 x 16. Finally expert with 99 mines and sizes of 16 x 30 or 30 x 16.

2 Session Two

2.1 Gauss Elimination Algorithm

Assuming that our base matrix has been transformed into the upper triangle from, thus, this other equation below it is denoted by the argument coefficient matrix shown in the equation. The component of the matrix A are not number of the original equation(excepting the 1st row) because has been mutated by reduction procedure, the same to the vector b

$$\left[\begin{array}{ccccccccc} A_{11} & A_{12} & A_{13} & \dots & A_{1k} & \dots & A_{1j} & \dots & A_{1n} \\ 0 & A_{12} & A_{13} & \dots & A_{2k} & \dots & A_{2j} & \dots & A_{2n} \\ 0 & 0 & A_{33} & \dots & A_{3k} & \dots & A_{3j} & \dots & A_{3n} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & 0 & 0 & & A_{kk} & & A_{kj} & & A_{kn} \\ \hline \vdots & \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & A_{ik} & \dots & A_{ij} & \dots & A_{in} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & A_{nk} & \dots & A_{nj} & \dots & A_{nn} \end{array}\right] \left[\begin{array}{c} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \\ \hline \vdots \\ b_i \\ \vdots \\ b_n \end{array}\right] \tag{M1}$$

When the ith row below the pivoting equation could be transformed, thus, the element A_{ir} must be eliminated. To have this, we can multiply the pivoting row by $\lambda = A_{ik}/A_{kk}$ from this row

$$A_{ij} \leftarrow A_{ij} - \lambda A_{kj}, \quad j = k, k+1, ..., n \tag{1}$$

$$b_i \leftarrow b_j - \lambda b_k \tag{1.1}$$

Transforming the coefficient of the matrix to upper from, k and l in 1, must be ranges of $k = 1, 2, ..., h-1$ (chooses the pivot row)

2.2 Back Substitution

$$\left[\begin{array}{ccccc} A_{11} & A_{12} & A_{13} & \dots & A_{1n} \\ 0 & A_{22} & A_{23} & \dots & A_{2n} \\ 0 & 0 & A_{33} & \dots & A_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & A_{nn} \end{array}\right] \left[\begin{array}{c} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{array}\right] \tag{M2}$$

$A_{nn}Xn$ is solved by

$$X_n = b_b/A_{nn} \tag{1.2}$$

The stage of back substation $X_n, X_{n-1}, ..., X_k + 1$ has been determinate, now we need to determinate X_k from Kth equation

$$A_{kk}X_k + A_{kk} + 1X_k + 1 + \cdots + A_{kn}X_n = b_k \tag{1.3}$$

The solution is

$$X_k = \left(b_k - \sum_{j=k+1}^{n} A_{kj}X_j\right) \frac{1}{A_{kk}}, K = n-1, n-2, \dots, 1 \tag{1.4}$$

Algorithm 1 Compute the Gauss Elimination

Require: $A \vee b$ {A is a matrix and b is the vector}
  $n \leftarrow$ length of b
  for k in range of 0 to $n-1$ **do**
    for i in range of $k+1$ to n **do**
      if $A_{i,k}$!= 0.0 **then**
        $\lambda \leftarrow A_{i,j}/A_{k,k}$
        $A_{i,k+1:n} \leftarrow A_{i,k+1:n} - \lambda A_{k,k+1:n}$
        $b_i \leftarrow b_i - \lambda b_k$
      end if
    end for
  end for
  for x in range of $n-1$ to -1, by -1 to -1 **do**
    $b_k \leftarrow (b_k - (A_{k,k+1:n} \cdot b_{k+1:n}))$
  end for
  return b {return the vector b }

2.3 Gauss Elimination Application

This technique is capable of identifying the number of free variables doing a reduction in the number of permutations needed to verify the solution. Gauss elimination is useful to solve system linear equations. It consists of two parts: the elimination phase and the solution phase as show table 1.1, the function to the elimination phase is to transform the equations into the form $U_x = c$.
To apply these methods, I chose the configuration showing the figure 1. All non-clicked squares

Table 1: Phases elimination.

	Initial form	Final form
Gauss Elimination	$A_x = b$	$U_x = c2$

are labeled x1 to x5. Each of them either contains a mine or it does not:

1	x_1
1	x_2
2	x_3
2	x_4
2	x_5

Figure 1: Game state study configuration.

Looking at the top left of figure above, show number one meaning the adjective top one. Then, looking for what number adjacent have the number one is 2: x_1 and x_2, therefore, is capable say that number of mines in x_1 and x_2 must add up to equal 1:

$$x_1 + x_2 = 1 \tag{1.5}$$

It let us say:

$$x_1 + x_2 = 1 \tag{2.3.1}$$
$$x_1 + x_2 + x_3 = 1 \tag{1.5}$$
$$x_2 + x_3 + x_4 = 2 \tag{2.1}$$
$$x_3 + x_4 + x_5 = 2 \tag{2.2}$$
$$x_4 + x_5 = 2 \tag{2.3}$$
$$\tag{2.4}$$

Therefore, that's equations would be written as

$$\left[\begin{array}{ccccc|c} 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 2 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right] \tag{M3}$$

and finally, applying the Gauss Elimination method:

$$\left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \tag{M4}$$

Negative numbers do not mean the Gauss Elimination has been failing, however, that's work perfectly giving us a partial solution of the vector x. Each non-clicked square in the minesweeper grid could just have a mine or not ($x \in [1, 0]$).

Dialing each row of the matrix above in a dot point:

- The first row should be able to denote that x_1 and x_5 not must be mines because $x_1 + x_5$ is not equal to 1,
- the second one has a negative number, which means that $x_1 - x_4 = 0$. This is true because x_2 is a mine and x_4 is not,
- The third one row had mines in x_3 and does not exist any more mines, therefore, is equal to 0.
- the four one row had mines in x_4 and x_5, both are mines because that is only to obtain 2.
- and the last one does not have mines.

Is possible to get more information from each row working at lower and upper bounds. In this case, there is only one configuration and would be applied to a row if it equal to an upper or lower bound, if it does not, multiple solutions are possible and probabilistic fields would emerge.

3 Session Three

3.1 NP-Complexity

The NP(non-deterministic polynomial time) is a complexity class used in computational complexity theory to classify decision problems. Which has proof in polynomial time by a deterministic Turing Machine.

In the NP class problem (all problems are solvable in polynomial time) is contained in NP; problems where his solution can be verified in polynomial time, thus, if the problems can be solved in polynomial time then his solution is also verifiable in polynomial time. However, NP contains more problems such as NP-complete. An algorithm that his resolution is in a polynomial-time must be solved also any other problem in polynomial time or P=NP? The most important NP problem.

3.2 Formal Definition

Corollary 1 *Begin Σ be a finite alphabet set and $\Sigma*$ be set of finite string over Σ, the language over Σ is and subset L of $\Sigma*$*

Deterministic machine M has an associate Σ. Each $W \in \Sigma^*$ exists a computation associated M with w. M accepts $\mathcal{W}$ if this computation accepts the state. The language is accepted by M, deleted $L(M)$.

$$L(M) = \{w \in \Sigma^* \ / \ M \ \ accept \ \ w\} \tag{2.5}$$

$Tm(w)$ is the number of steps in the computation of M on w. If the computation never halts, $Tm(w) = \infty$. For $n \in \mathbb{N}$, $Tm(h)$ worst-case return

$$Tm(n) = \ max\{Tm(w) \ / \ w \in \Sigma \ \ n\} \tag{3.2.1}$$

$$\Sigma^n C\Sigma = \{All \ \ string \ \ over \ \ \Sigma \ \ of \ \ length \ \ n\} \tag{3.2.2}$$

This M runs in a polynomial time if $\exists k \forall n, Tm(n) \leqslant n^k + k$, thus

$$P = \{L \ / \ L = L(m)\} \tag{3.2.3}$$

For and Σ_1, we have $R \subseteq \Sigma^*$ x Σ_1^* when each case R in a language LR over $\Sigma U \Sigma_1 U \{\#\}$

$$LR = \{w\#y/R(w,y)\}, \tag{3.2.4}$$

when

- $\#$ is not in Σ
- R is polynomial-time if $LR \in P$

NP languages by L over Σ in NP $\exists k(k \in N)$ and a polynomial time in R $\exists w \forall w(w \in k^*)$.

$$wwL \longleftrightarrow \exists y(|y| \leqslant |w|^k \ \ and \ \ R(w,y)) \ \ when \ \ |w|, |y| = w, y \tag{3.2.5}$$

3.3 Minesweeper Complexity

This game has been different in complexity, even in electrical engineering. Richard Kaye in his paper called 'Minesweeper is NP-Complete', creates a wire construct and Boolean logic gates from minesweeper configurations. Proving a polynomial-time reduction. In another way, Allan Scott in 'Minesweeper may not be NP-Complete but it had nonetheless' asks for there exists at least one covered square that his contents can be safely inferred, given us a consistent configuration to get this result Allan shows that the consistent configuration is CO-NP. One set of two consistent boards for covered squares(S is a mine and one where is not a mine). Proving that a deterministic consistency cannot be made since each covered square.

Iterating through each covered square s in each possible board and s be consistent in eight neighbors. Given a board nxm, O(n$^2m^2$)$timetakespolynomialtime$.

Kaye's proof proves a similar approach as Allan defines circuits: Allan creates wires, logical operators (AND and NOT gates), and terminals allow creating any Boolean circuits.

References

[1] *Mastering the ancient game of Go with Machine Learning, Google DeepMind.* David Silver and Demis Hassabis, 2016.
`http://ai.googleblog.com/2016/01/alphago-mastering-ancient-game-of-go.html`.

[2] *Numerical Methods in Engineering with Python.* Jan Kiusala, 34-36, 2016.

[3] *Algorithm Design.* Jon Klinberg, Eva Tardo, 464, 2006.

[4] *THE P VERSUS NP PROBLEM*, Stephen Cook
`https://claymath.org/sites/default/files/pvsnp.pdf`