Machine Learning Applications To Minesweeper Solver

Li Derun
School of Science and Information
ShanghaiTech University
Pudong new area, Shanghai
`lidr@shanghaitech.edu.cn`

Zhang Cenyang
School of Science and Information
ShanghaiTech University
Pudong new area, Shanghai
`zhangcy1@shanghaitech.edu.cn`

Weng Yijie
School of Science and Information
ShanghaiTech University
Pudong new area, Shanghai
`wengyj@shanghaitech.edu.cn`

Abstract

This project aims at training a minesweeper solver using machine learning knowledge learned in SI151. Three algorithm (linear classification, Q-learning and neural network) are designed, trained and tested, among which BPNN shows the best performance. To improve BPNN applying to larger game complexity, concept of convolution is used and results in a considerable winning rate improvement.

Keywords: minesweeper; machine learning; neural network

1 Introduction

1.1 Minesweeper game

Minesweeper is a popular game around the world developed by Microsoft in 1980s. Player can click the hidden squares to uncover them. If the clicked square is not a mine, it reveals an integer which denotes the number of adjacent tiles with mines. Players can deduce which squares contain mines according to the limited information, and when all squares without mines are uncovered, players win the game.

Minesweeper also hides in itself one of the most important problems of mathematics: whether P=NP. Moreover, it has been proved by Kanye, R (2000) that minesweeper is a NP-complete problem and belongs to a more difficult class of problems called sharping P-complete. Therefore, designing a minesweeper solver by conventional math algorithms could be complicated.

1.2 Minesweeper with machine learning

Consider that conventional algorithm of minesweeper solver could be complex, some approaches have been made to solve games using machine learning. In 2015, Luis Gardea, Griffin Koontz, Ryan Silva build two minesweeper solvers by supervised learning and simplified Q-learning. However, the win rate of solver drops sharply as the board size maximizes. In 2017, Jacob J.Hasen et.al compared Q-learning with other two algorithms. Moreover, there are also some people trying to solve minesweeper by convolutional neural network and getting relatively good results.

Preprint. Under review.

2 Methodology and Experiment

After discussion and comparison, three types of algorithm come up, which are linear classification, Q-like learning(based on reinforcement learning) and back propagation neural network. A simple minesweeper model is used instead of professional minesweeper game to simulate game state from reference[3].

2.1 Naive linear classification

2.1.1 Algorithm

First, we build a simple solver by naive linear classification. This algorithm holds an assumption that covered squares could be classified by a linear model. Input X indicates states of the 8 neighbors to the considered square and the output Y is a variable indicating its probability to be a mine. The output is effected locally by the numbers shown at its neighbors. Linear assumption gives the expression of estimator by $\hat{y} = x^T w$. When trainning the model, we estimate probabilities of every unrevealed tile to be a mine, and choose one with the least probability to unreveal. If unrevaling a mine, increase more relative coefficients and decrease others. After initializing coefficients by random normal distribution, they are updated as training game continues. The algorithm is shown as below:

Algorithm 1 Naive classification

Input: $X^T = [x_1, x_2, \cdots, x_8]$ (eight neighbors), with $x_i = \{-1, 1, 2, \cdots, 8\}$(-1 for unrevealed tiles).
Output: category $G = \{0, 1\}$.(0 for safe and 1 for a mine)

```
 1: Assume $X \leftarrow \begin{bmatrix} 1 \\ X \end{bmatrix}$, $\mathbf{w}^T = [w_0, w_1, \cdots, w_8]$
 2: Begin by probing a corner square
 3: while $w_i$ not converge do
 4:     while not game over do
 5:         for tile on board do
 6:             probability $y \leftarrow \mathbf{w}^T X$
 7:         end for
 8:         Unreveal the tile with minimum y
 9:     end while
10:     $w_i \leftarrow w_i + \frac{1}{1+\exp(-x)} * (1 - \frac{1}{1+\exp(-x)})$
11: end while
```

2.1.2 Result

Although its win rate is larger than random choose, the result of naive linear classification is extremely poor and weights can hardly converge. Considering that the assumption of algorithm (the state of squares could be linear separate) is really robust, it is reasonable that the performance is really bad.

2.2 Q-like-learning

2.2.1 Algorithm

Q-learning is a reinforcement learning algorithm aiming at discovering the best action for each given board configuration, which models minesweeper as an MDP. After learning, the algorithm uses the values obtained to play. In 2015, Luis Gardea et al. improved the Q-learning algorithm and applied it to minesweeper. The algorithm is shown in the following column.
In this algorithm, only correct moves are considered. Substituding Q value for a given (state,action) pair, $\hat{P}(s, a)$ resembles the probability that action a will not cause failure. Given n visits to state s, the closer $\hat{P}(s, a)$ is to n, the less likely the corresponding tile contains mine. This assumes that s is large enough to acquire a proper probability distribution.

Algorithm 2 improved Q-Learning

```
1: Begin by probing a corner square
2: while not game over do
3:     s← current state of the board
4:     $Array$ ← all tiles on frontier not mines
5:     for tile in $Array$ do
6:         $\hat{P}(s,a) \leftarrow \hat{P}(s,a) + 1$
7:     end for
8: end while
```

Winning rate of this algorithm decreases sharply as the board size increases. For example, when board size extracts from 4×4 to 5×5, the state space will increase by a factor $f \sim \frac{10^{25}}{10^{16}} = 10^9$. To overcome this drawback, we develop a new algorithm called Q-like-learning to reduce the complexity. The new algorithm is shown below:

Algorithm 3 Q-like-learning

```
1: Begin by probing a corner square
2: while not game over do
3:     s← current $3 \times 3$ board of current square and location
4:     $Array$ ← all tiles on frontier not mines
5:     $\hat{P}(s,a) \leftarrow \hat{P}(s,a) + 1$
6:     Board.append(s)
7: end while
```

After discarding some insufficiently useful information (focusing more on local information), the complexity decreases dramatically. For example, the complexity of original algorithm is $O(10^n)$, where n denotes the number of tiles in board, but the improved one is only $O(10^8 n)$.
This reduction of complexity greatly optimizes the original Q-learning method.

2.2.2 Result

Theoretically, the Q-like-learning algorithm could perform much better than Q-learning. However, limited by poor hardware conditions, the amount of parameters is still too large for us to get a good experiment result.

2.3 Neural network

2.3.1 Algorithm

Based on the experience above and knowledge about Neural Network learned in class, we try to apply Back Propagation Neural Network to Minesweeper problem. Considering the global information, we define the input layer as the state of each square (covered or numbered from 0 to 8) along with the radio between numbers of mines and remaining uncovered tiles, such that the size of input layer is $N =$ number of tiles $* 10 + 1$. Meanwhile, the output layer indicates the probability of each of m squares to be mine. According to Huang et al.(2003), one way to define an optimal size of hidden layer is $\sqrt{(m+2) * N} + 2\sqrt{N/(m+2)}$. Since it is a BPNN with only one hidden layer, conventional sigmoid activation function $\sigma(x) = \frac{1}{1-e^x}$ is used.
For initialization, according to Sakesh P., Xavier Glorot normal initialization of weight is suitable for sigmiod activation function. We also choose to initialize the bias to be 0. At each single step of the training game, all weights and bias are updated by gradient descent with the true mine distribution as the true value of output.

2.3.2 Result

We test BPNN model on different board sizes with 4000 rounds of training, and the output is Figure 1. From the figure we can see that our model trains 4x4 board with 3 mines well. Expanding the training set to 50000 games, the result is obtained in Figure 2. As seen, good convergence can be

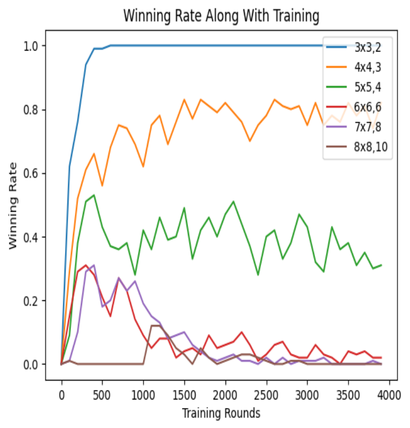

Figure 1: winning rate along with training

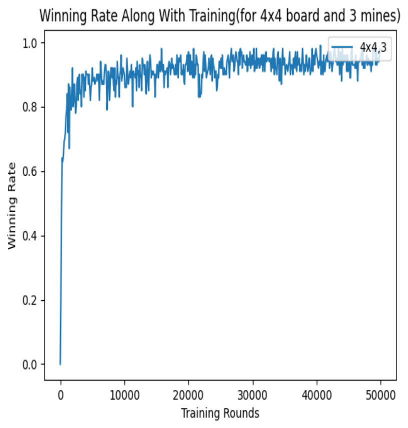

Figure 2: winning rate along with traning(for 4×4 board and 3 mines

Table 1: average testing winning rates in 1000 rounds

Game model	Average winning rate	Game model	Average winning rate
4x4 board, 3 mines	84.29%	6x6 board, 6 mines	4.29%
5x5 board, 3 mines	57.27%	7x7 board, 8 mines	1.07%
5x5 board, 4 mines	26.50%	8x8 board, 10 mines	0.35%

achieved using BPNN model, and the winning rate successfully reaches a high level around 90%. However, we desire to apply the model to larger sizes of minesweeper game. Based on the concept of convolution, the board is masked by 4x4 squares and the probabilities are calculated by averaging the outputs of each 4x4 square, using BPNN trained 100000 times with random number of mines from 2 to 4. The output of the new algorithm is shown in Table 1. The winning rate has an obvious improvement comparing to directly training corresponding game models (larger than 5x5 hardly has a win if directly trained). Nevertheless, the winning rate is still quite low as size increses, which is considered as the result of underfitting the too large amount of parameters (over 10^{n^2} game state cases). Simple BPNN learned in class and insuffient hardware capacity are quite stricted, but we suppose that Convolutional Neural Network can have better performance. Actually, Ryan B. solved this problem by CNN with satisfying winning rate. However, CNN does not perform such well in small scale game model due to overfitting.

3 Conculsions

In this project, some attemps were made to find a minesweeper solver with machine learning methods which could have a good winning rate with comparatively low complexity. With this belief, three algorithms are used. The first is a naive linear classifier. Due to its robust assumption, it holds a really low complexity but poor performance while training. Secondly, improving traditional Q-learning, Q-like-learning is developed. However, limited by laptop hardware, experiment on model with code cannot be conducted. Finally, we realize a back propagation neural network model and train it on 4×4 board. It shows good properties of convergence and acquires a relatively good winning rate around 90%. The model can be expanded to convolution neural network by masking the well-trained 4×4 model on larger game sizes and achieve pretty good results.

Our models have great potential to hold much better performance. It can be improved by adjusting parameters to get a better network structure or applying an advanced convolutional neural network model. Also, we could try to implement the Q-like-learning algorithm in a better way on stronger hardwares to verify our theoretical analysis.

References

[1] Luis, G & Griffin, K (2015) Train a minesweeper solver. CS229.

[2] Kanye, R (2000) Minesweeper is NP-complete *The Mathematical Intelligencer, 22, 9-15.*

[3] Project webpage. *https://github.com/ryanbaldini/MineSweeperNeuralNet*

[4] Project webpage. *https://medium.com/@sakeshpusuluri123/activation-functions-and-weight-initialization-in-deep-learning-ebc326e62a5c*