Minesweeper (D)Q-Agent

Papadopoulos Argiris

Technical University of Crete

Contents

Introduction

- Reinforcement Learning is widely known for its application in games and apps
- An agent is trained by taking actions in an environment and learns through trial and error

- Q-Learning is mostly used for problems like the one we are going to tackle and can be paired well with Deep Neural Networks.

Minesweeper Game

- Open all non-bomb cells to win
- Lose by revealing a bomb
- A cell with a number indicates the number of bombs adjacent

Agent-Environment Communication

- Environment: Classic Windows XP Minesweeper.exe
- Python scripting
- Interaction by utilizing pynput for automated mouse-keyboard use and mss for capturing screenshots

Q-Agent

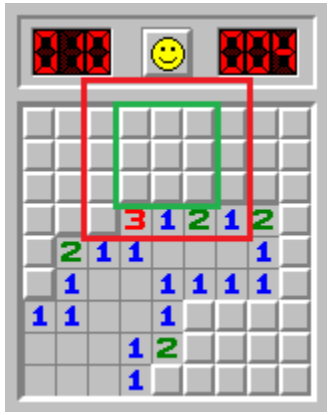
- A Beginner Mode Board is a 9x9 dimension cell grid.
- Total different states 11^{81}(*) with 81 actions (each action corresponds to a cell)
- Impossible to solve!

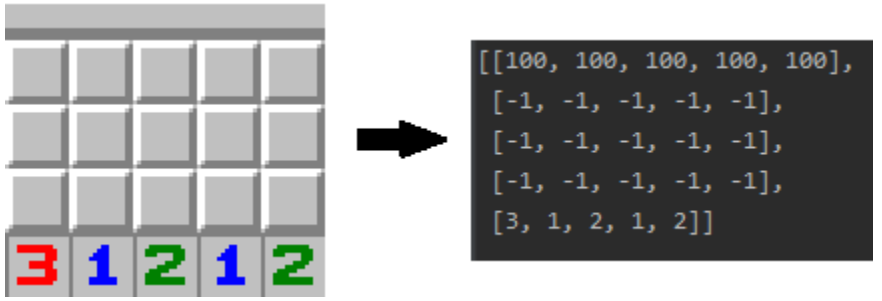
(*)Possible values for a cell (total 11):

- 1-8 number
- Bomb
- Unknown cell
- Empty open cell

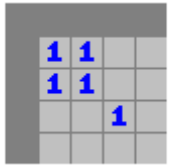
Q-Agent (1)

- States: 5x5 grid
- Actions: 9, on 3x3 grid (0: top-left, 1:top-mid, 5:mid-right)
- Choose each 3x3 grid ONCE but cover whole Board
- In 9x9 Beginner we have a total of 9 states (9 3x3 grids)





- Translate image to list of 5x5
- Still 11^{25} is high, but plenty of states don't exist, i.e.:



Q-Agent (2)

- Select max $Q(s,a)$ for every state, action and hit the respective cell
- Initialize $Q(s,a) = 0$ for each action every time an unknown state is read
- On unclickable cells (already open) set $Q(s,a) = -\infty$
- Update Q-table every time an action is taken:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a' \in A} Q(s',a') - Q(s,a) \right)$$

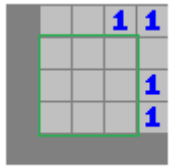
- By revealing a bomb or if the new 5x5 state doesn't contain any clickable cell update by $Q(s,a) \leftarrow Q(s,a) + \alpha \left(r - Q(s,a) \right)$

Q-Agent (3)

- By revealing a bomb $reward = -5$
- Otherwise $reward = +1$

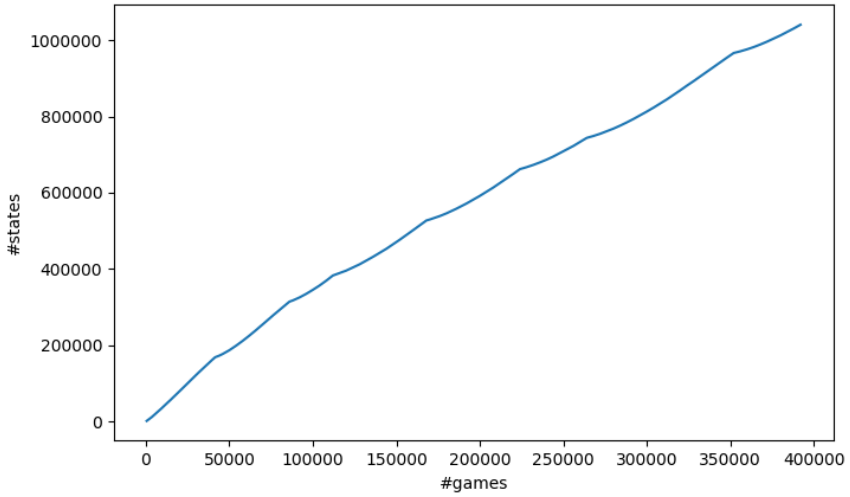
- Utilize $epsilon$ and $decay = 0.999975$ to enable exploration

- Ignore states (reduce table length) which the agent cannot take an action upon (no unknown cell in center 3x3) i.e.:

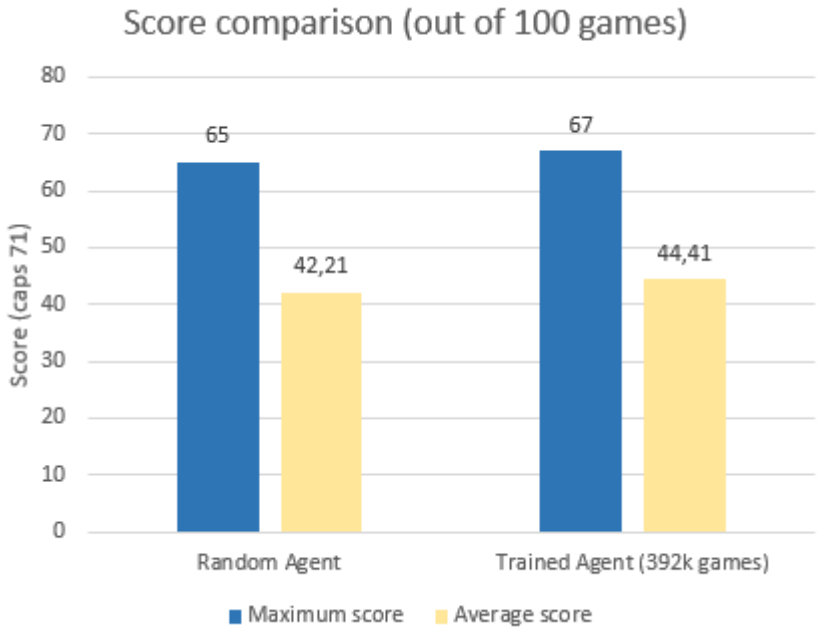


Q-Agent (4) - Results

- Agent trained for 392.000 games (approx. 163 hours)
- Discovered 1.040.292 different states
- State discovery didn’t converge to 0 -> requires further training



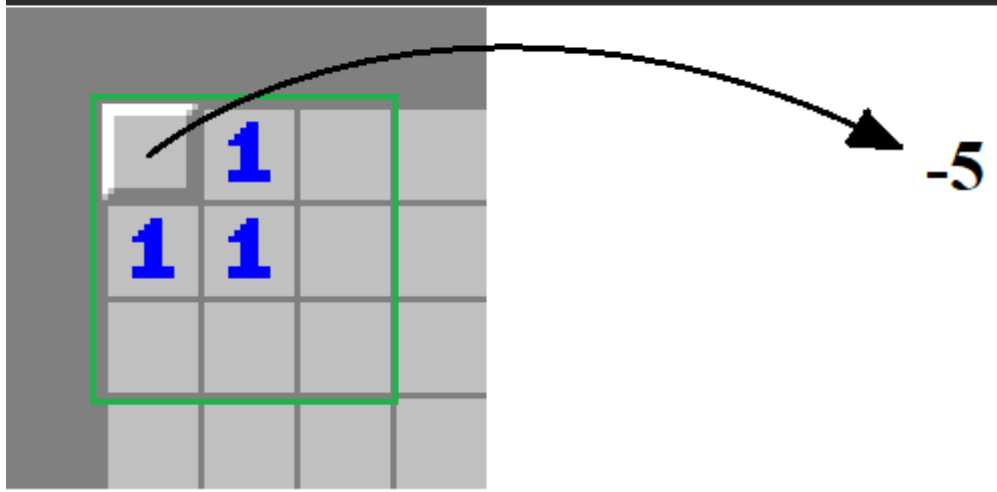
Q-Agent (5) - Results



- Trained Agent performs like a Random Agent, score is not a valid metric yet…
- Score caps at 71 for Beginner Mode

Q-Agent (6) - Example

```
state = str([[100, 100, 100, 100, 100], [100, -1, 1, 0, 0], [100, 1, 1, 0, 0], [100, 0, 0, 0, 0], [100, 0, 0, 0, 0]])
Qtable[state]
: [-5.0, -inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf]
```



- Selected unknown cell is certainly a bomb
- Action 0 has -5 Q-value, agent correctly won’t choose this action (other undiscovered actions will have Q-value = 0)

Deep Q-Agent

- Neural Network Architecture (Keras):
  - Convolutional 2D layer, image input
  - MaxPooling2D layer, for downscaling images
  - Flatten, reduces dimensions of a list i.e.: 64x32x32 becomes 65.536x1
  - Dense layer, typical neural network layer

- Input of Network: images with dimension $(9 \cdot 16) \times (9 \cdot 16) \times 1$ width, height of image (9 cells 16 pixels each) and 1 channel
- Output of Network: 81 Q-values corresponding to each action

Deep Q-Agent

- This wasn’t trained due to processing power limitations, each game took around 20-30 seconds to run.

Conclusion

- By dividing the board to individual states the problem can be tackled and solved.
- Also the use of Neural Networks and Convolutional layers provide a reliable solution.

- Future tasks:
  - Train Q-Agent until state discovery converges to 0
  - Run other versions of Q-Agent
  - Install GPU version of Python for efficient Neural Network Training

Thank you for your time

Python CPU version 3.7:

- pynput
- time
- datetime
- Keyboard
- mss
- Random
- Numpy
- Pickle
- sys
- Tensorflow
- Keras

Sources:

- https://pythonprogramming.net/
- https://www.geeksforgeeks.org/
- https://docs.python.org/3/