AI Games course

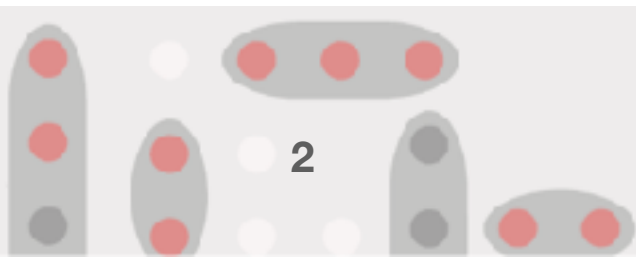Certificate 1, session 3
"What does the data say?"

AIgaming.com

Strategy so far

- In the *target* mode, search through the space around the wounded piece of ship.

- In the *hunt* mode, aim randomly.

AIgaming.com

Strategy so far

- In the *target* mode, search through the space around the wounded piece of ship.

- In the *hunt* mode, aim randomly.

Let's improve on that!

AIgaming.com

Learning from data

- **Idea**: instead of randomly searching for a ship, let's *predict* where it *probably* is.

- there may be a pattern in how a user arranges ships, e.g.:
  - all ships stacked in one place, or
  - all ships in corners, etc.

- given what you already know about the opponent's board, hit the cell that is *most likely* to contain a ship, *based on the data.*
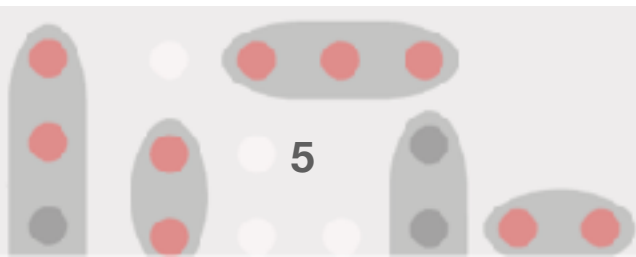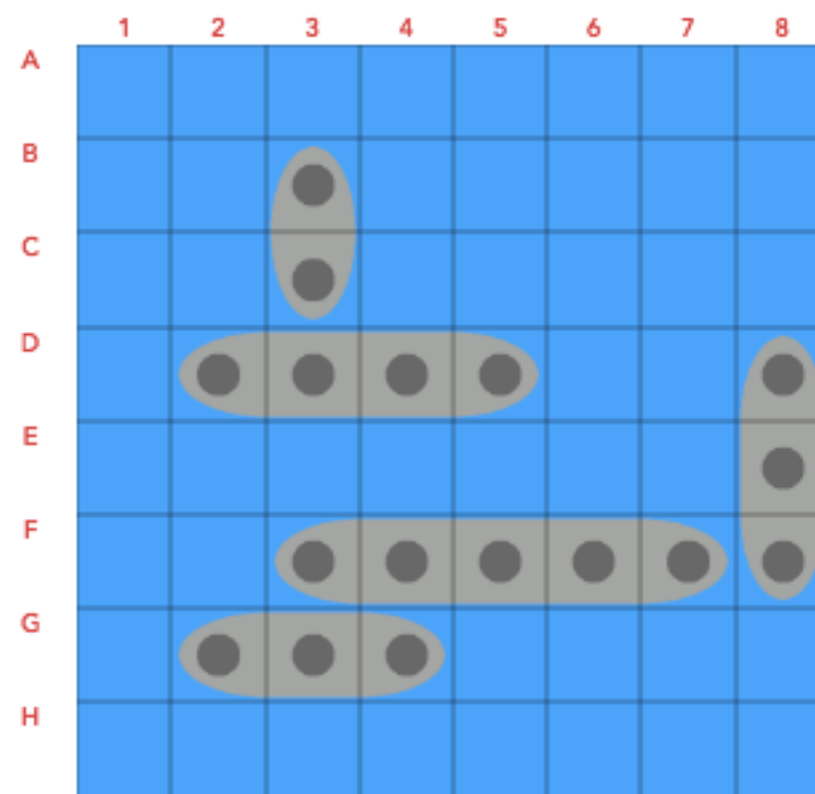
AIgaming.com

User logs

- download *boards.txt*

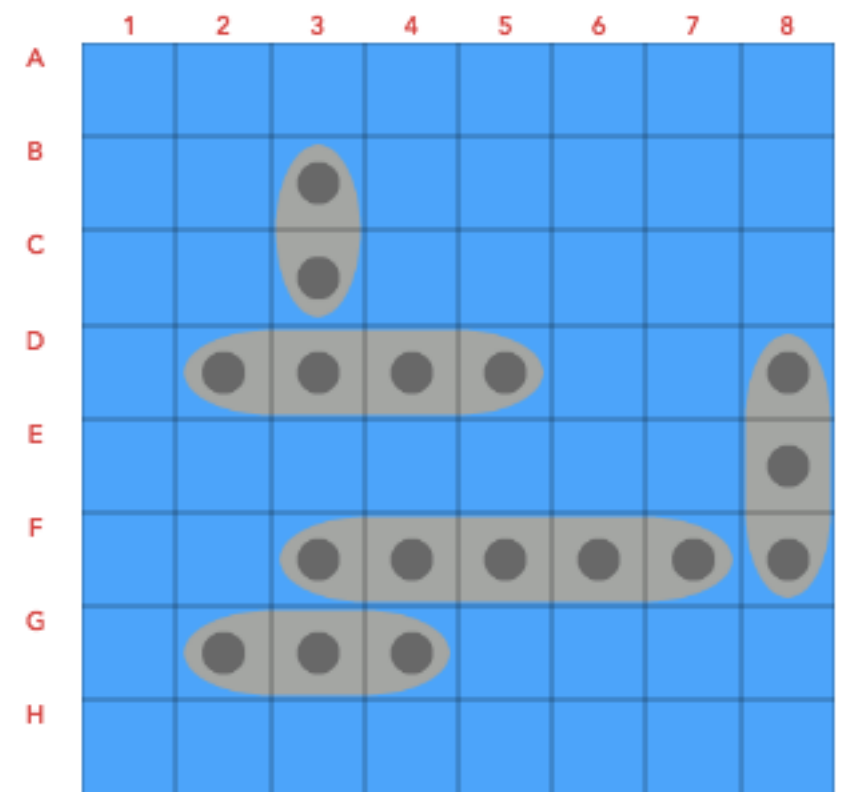- 60000+ real board arrangements generated by the users (only 179 default arrangements)

AIgaming.com

User logs

- download *boards.txt*

- 60000+ real board arrangements generated by the users (only 179 default arrangements)

```python
file = open('boards.txt', 'r')
boards_set = file.read()
boards = boards_set.splitlines()
print(len(boards)) # 67174 boards
```

```
…
[['', '', '', '', '', '2', '', '0'], ['', '4', '', '3', '', '2', '', '0'], ['', '4', '', '3', '', '2', '', '0'], ['', '', '', '3', '',
'', '', '0'], ['', '', '', '', '', '1', '', '0'], ['', '', '', '', '', '1', '', ''], ['', '', '', '', '', '1', '', ''], ['', '', '', '', '',
'1', '', '']]
[['', '', '', '', '', '', '', ''], ['', '', '', '', '', '', '', ''], ['', '', '', '', '', '', '', '1'], ['', '4', '4', '', '', '2', '', '1'],
['', '', '', '', '', '2', '', '1'], ['', '3', '3', '3', '', '2', '', '1'], ['', '', '', '', '', '', '', ''], ['', '0', '0', '0', '0',
'0', '', '']]
[['', '', '', '', '', '', '', ''], ['', '', '', '', '', '', '', ''], ['1', '1', '1', '1', '', '', '', '3'], ['', '', '', '', '', '', '', '',
'3'], ['', '', '', '', '', '', '', '3'], ['2', '2', '2', '', '', '4', '', ''], ['', '', '', '', '', '4', '', ''], ['0', '0', '0', '0',
'0', '', '', '']]
…
```
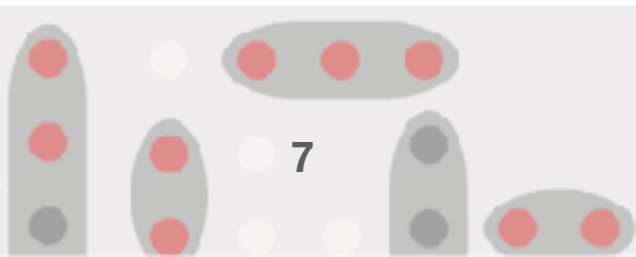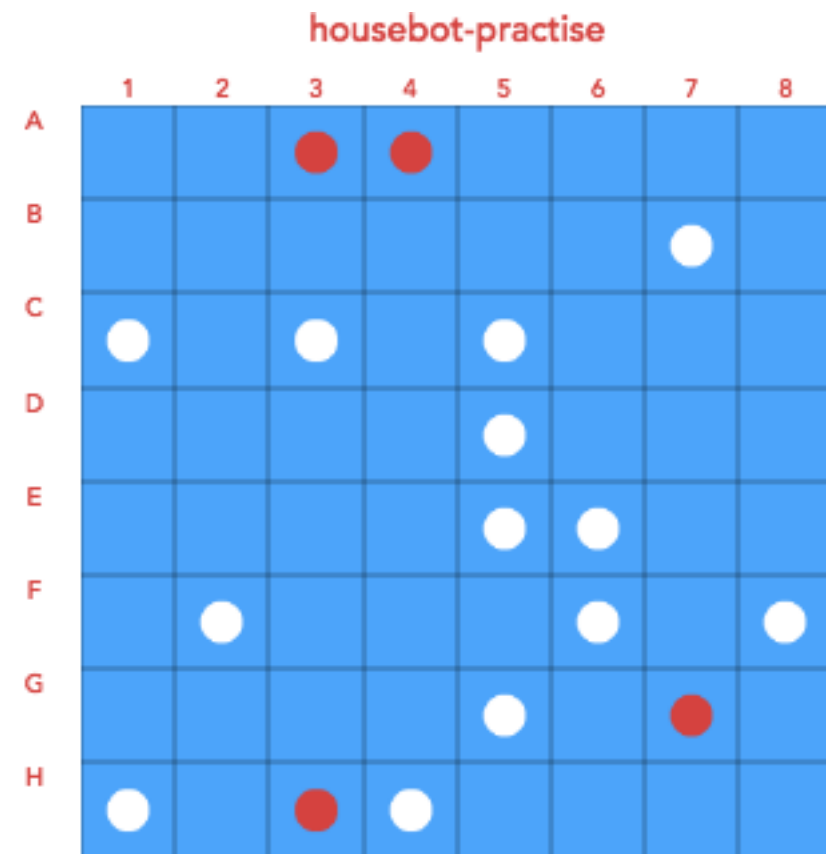
Predicting ship placement

1. from *gameState*, parse the current state of the opponent's board:

'OppBoard': [['', '', 'H', 'H', '', '', '', ''], ['', '', '', '', '', '', 'M', ''], ['M', '', 'M', '', 'M', '', '', ''], ['', '', '', '', 'M', '', '', ''], ['', '', '', '', 'M', 'M', '', ''], ['', 'M', '', '', '', 'M', '', 'M'], ['', '', '', '', 'M', '', 'H', ''], ['M', '', 'H', 'M', '', '', '', '']]

2. in *boards.txt*, find those boards that coincide with *OppBoard* on the known cells:

E.g., [['0', '0', '0', '0', '0', '', '', '3'], ['', '', '', '', '', '', '', '3'], ['', '', '', '', '', '', '', '3'], ['2', '2', '2', '2', '', '', '', ''], ['', '', '', '', '', '', '1', ''], ['', '', '', '', '', '', '1', ''], ['', '', '', '', '', '', '1', ''], ['', '4', '4', '', '', '', '1', '']]

We will call them *similar boards.*
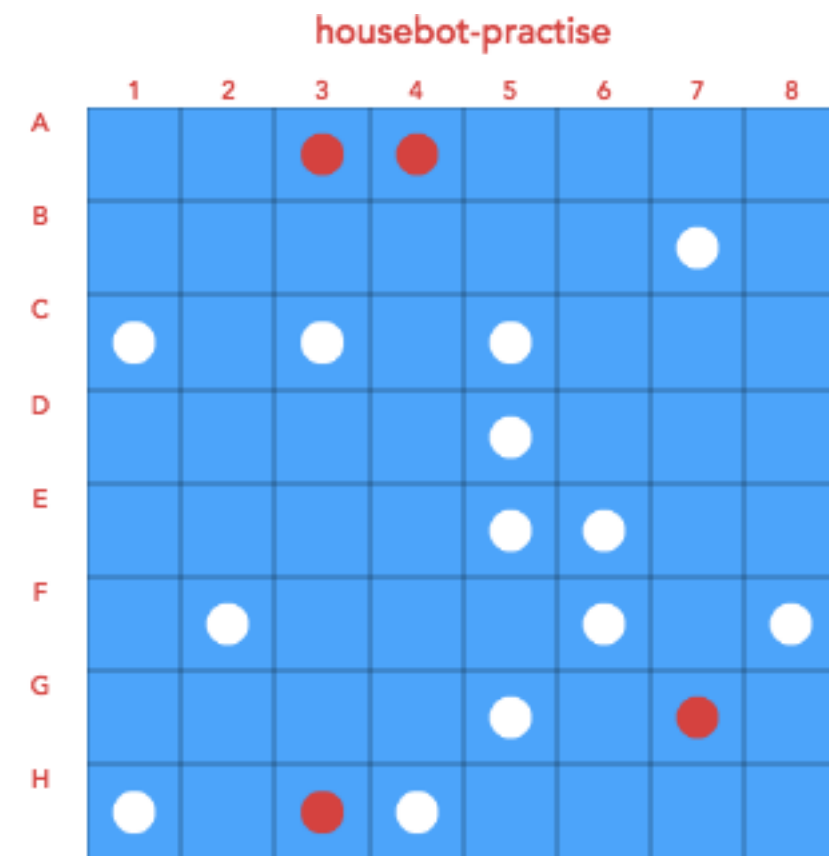

housebot-practise

AIgaming.com

Predicting ship placement

3. find cells on similar boards that:

- are not yet opened on *OppBoard*, and

- contain ships in the maximum number of similar boards.

E.g.: [['', '', 'H', 'H', '', '', '', ''], ['', '', '', '', '', '', 'M', ''], ['M', '', 'M', '', 'M', '', '', ''], ['', '', '', '', 'M', '', '', ''], ['', '', '', '', 'M', 'M', '', ''], ['', 'M', '', '', '', 'M', '', 'M'], ['', '', '', '', 'M', '', 'H', ''], ['M', '', 'H', 'M', '', '', '', '']]

housebot-practise

Predicting ship placement

for every unopened cell on *OppBoard*:

    cell_score := # of similar boards that contain ship in that cell;

pick the cell with the highest score.

AIgaming.com

Predicting ship placement

▸ **battleships_move.py:**

    ▸ function *calculateMove*

```python
4
5    def calculateMove(gamestate):
6        if gamestate["Round"] == 0:  # If we are in the ship placement round
7            # move = exampleShipPlacement()  # Does not take land into account
8            move = deployRandomly(gamestate)  # Randomly place your ships
9        else:  # If we are in the ship hunting round
10           move = chooseRandomValidTarget(gamestate)  # Randomly fire at valid sea targets
11       return move
12
```

Let's modify the *else*-statement to accommodate probabilities.

AIgaming.com
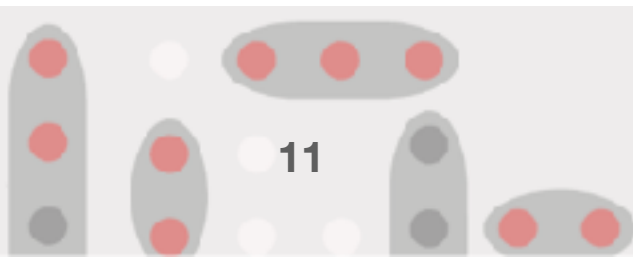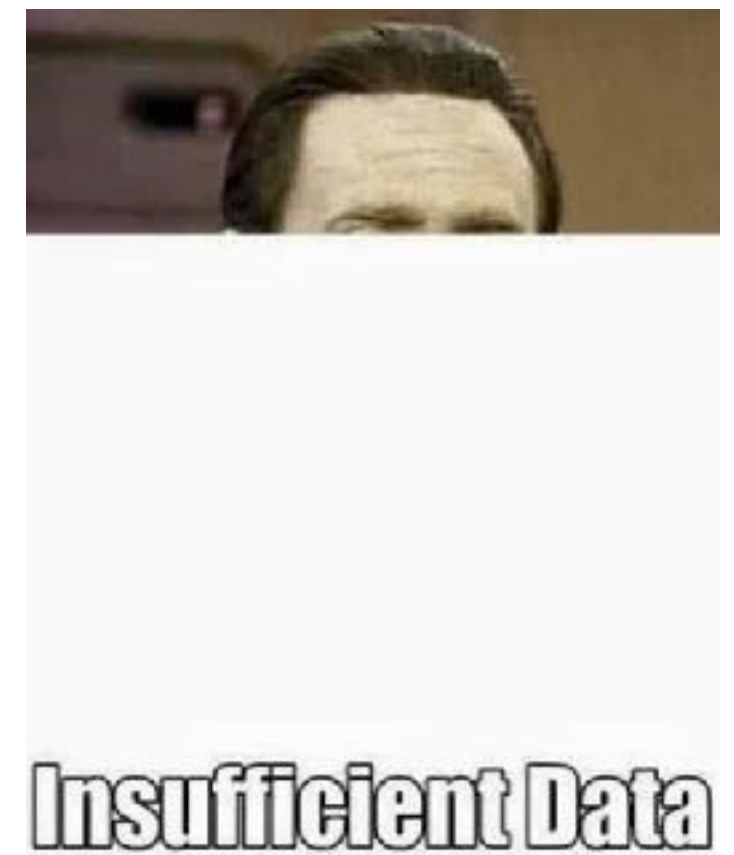
Handling data sparsity

- 60,000 boards is not that much data to learn from!

- what if you cannot find similar boards?

    Ex 1: round 3, thousands of similar boards that do not contain a ship in (row B, column 7)

    'OppBoard': [['', '', '', '', '', '', '', ''], ['', '', '', '', '', '', 'M', ''], ['', '', '', '', '', '', '', ''], ['', '', '', '', '', '', '', ''], ['', '', '', '', '', '', '', ''], ['', '', '', '', '', '', '', ''],['', '', '', '', '', '', '', ''], ['', '', '', '', '', '', '', '']]
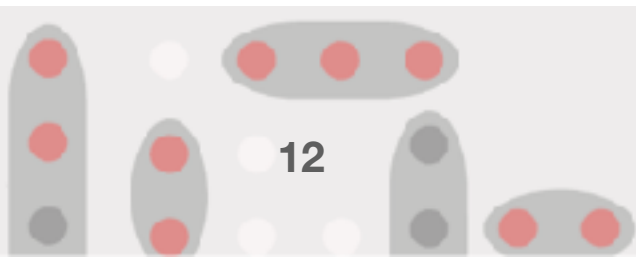
    Ex 2: round 30+, possibly no similar boards with the given ship arrangement

    E.g., [['H', 'H', 'H', 'H', 'M', '', '', 'H'], ['M', 'M', '', 'M', '', '', '', 'H'], ['M', 'M', '', '', 'M', '', '', ''], ['H', 'M', 'M', 'H', 'M', '', 'M', ''], ['M', 'M', 'M', '', 'M', 'M', 'H', ''], ['M', '', 'M', 'M', '', 'M', 'H', 'M'], ['M', 'M', 'M', 'M', '', '', 'H', ''], ['', 'H', 'M', '', '', 'M', 'H', '']]

AIgaming.com

Handling data sparsity

- in case there are no similar boards for a given OppBoard:

  ‣ either revert to *deployRandomly(gamestate)*, or

  ‣ make a *soft comparison* of boards, i.e., find boards that are most similar to OppBoard

AIgaming.com