

Build an Adversarial Game Playing Agent

Experiment 2: Develop an opening book

1. Algorithm for opening book:

- Describe your process for collecting statistics to build your opening book. How did you choose states to sample? And how did you perform rollouts to determine a winner?

For the first step, I choose a loop to run every choice from position 0 to 114 (for a row9 x column11 board), for the subsequent steps, I used random choice to expand depth pile to 4, then I run simulation to complete the game using “random choice” to pick up the actions for each player, and return -1 if the active player lose or +1 if the active player win. I accumulate the above reward score (1 or -1) by running 20000 simulation games, and for each board state, return the action that will get the highest accumulated reward score, to build the opening book dictionary.

- What opening moves does your book suggest are most effective on an empty board for player 1 and what is player 2's best reply?

My opening book return the corner moves 114 is the best, and the player 2 replied 61 as the best response.

2. Script for the opening book:

```
import pickle
import random
from isolation import Isolation
from collections import defaultdict, Counter

NUM_ROUNDS = 20000
size = 115

def build_table(num_rounds=NUM_ROUNDS):
    # Builds a table that maps from game state -> action
    # by choosing the action that accumulates the most
    # wins for the active player. (Note that this uses
    # raw win counts, which are a poor statistic to
    # estimate the value of an action; better statistics
    # exist.)

    book = defaultdict(Counter)

    state = Isolation()
    initial_moves = state.actions()
    for x in range(num_rounds):
        if x % 200 == 0:
            print(x)

        state = Isolation()
        for i in initial_moves:
            book[state][i] += build_tree(state.result(i), book, depth=2)
            state = Isolation()
```

```

openbook = {k: max(v, key=v.get) for k, v in book.items()}
return openbook

def build_tree(state, book, depth):
    if depth <= 0 or state.terminal_test():
        return simulate(state)
    action = random.choice(state.actions())
    reward = build_tree(state.result(action), book, depth - 1)
    book[state][action] += reward
    return reward

def simulate(state):
    player_id = state.player()
    while not state.terminal_test():
        state = state.result(random.choice(state.actions()))
    return -1 if state.utility(player_id) < 0 else 1

book = build_table()

f = open("data20000.txt", 'w')

with open("data.pickle", 'wb') as f:
    pickle.dump(book, f)

```

3. Experiment using opening book:

Using the opening book for the first 4 steps in CustomPlayer:

Shunlings-MacBook-Pro:3_Adversarial_Search shunling\$ python3 run_match.py -r 40 -o RANDOM -p 4

Running 80 games:

```

+++++
+++++

```

Your agent won 93.8% of matches against Random Agent

Shunlings-MacBook-Pro:3_Adversarial_Search shunling\$ python3 run_match.py -r 40 -o GREEDY -p 4

Running 80 games:

```

--++
++

```

Your agent won 50.0% of matches against Greedy Agent

Shunlings-MacBook-Pro:3_Adversarial_Search shunling\$ python3 run_match.py -r 40 -o MINIMAX -p 4

Running 80 games:

```

++
+

```

Your agent won 45.0% of matches against Minimax Agent

Using open book for the first 3 steps CustomPlayer:

```
Shunlings-MacBook-Pro:3_Adversarial_Search shunling$ python3 run_match.py -r 40 -
o RANDOM -p 4
```

Running 80 games:

```
+++++
+++++
```

Your agent won 96.2% of matches against Random Agent

```
Shunlings-MacBook-Pro:3_Adversarial_Search shunling$ python3 run_match.py -r 40 -
o GREEDY -p 4
```

Running 80 games:

```
+++++
+++++
```

Your agent won 100.0% of matches against Greedy Agent

```
Shunlings-MacBook-Pro:3_Adversarial_Search shunling$ python3 run_match.py -r 40 -
o MINIMAX -p 4
```

Running 80 games:

```
+++++
+---+
```

Your agent won 62.5% of matches against Minimax Agent

Using open book for the first 2 steps:

```
Shunlings-MacBook-Pro:3_Adversarial_Search shunling$ python3 run_match.py -r 40 -
o RANDOM -p 4
```

Running 80 games:

```
+++++
+++++
```

Your agent won 95.0% of matches against Random Agent

```
Shunlings-MacBook-Pro:3_Adversarial_Search shunling$ python3 run_match.py -r 40 -
o GREEDY -p 4
```

Running 80 games:

```
+++++
+++++
```

Your agent won 100.0% of matches against Greedy Agent

```
Shunlings-MacBook-Pro:3_Adversarial_Search shunling$ python3 run_match.py -r 40 -
o MINIMAX -p 4
```

Running 80 games:

```
+-----
+++++
```

Your agent won 60.0% of matches against Minimax Agent

Using open book only for the first step:

```
Shunlings-MacBook-Pro:3_Adversarial_Search shunling$ python3 run_match.py -r 40 -o RANDOM -p 4
```

Running 80 games:

```
+++++
+++++
```

Your agent won 93.8% of matches against Random Agent

```
Shunlings-MacBook-Pro:3_Adversarial_Search shunling$ python3 run_match.py -r 40 -o GREEDY -p 4
```

Running 80 games:

```
+++++
+++++
```

Your agent won 100.0% of matches against Greedy Agent

```
Shunlings-MacBook-Pro:3_Adversarial_Search shunling$ python3 run_match.py -r 40 -o MINIMAX -p 4
```

Running 80 games:

```
+++++
++--
```

Your agent won 48.8% of matches against Minimax Agent

Summary Table of Winning rates against three opponents with opening book for selection of the first steps:

	Random	Greedy	MiniMax
First step	93.8%	100%	48.8%
First 2 steps	95%	100%	60%
First 3 steps	96.2%	100%	62.5%
First 4 steps	93.8%	50%	45%

Conclusion: Use opening book for the first three steps is the best.