

1 MCTS Primer: FlatMCSearch

Here, we will be implementing (together) a class called `FlatMCSearch`, which is a simplified version of 2-player Monte Carlo Tree Search. You should fill in the blank lines, denoted by blanks that look like this:

.....

Make sure to read the comments which describe what the lines are doing. These questions will help you understand the interface of our AlphaZero assignment (HW12).

1.a rollout_uniform

Before we can implement the full `FlatMCSearch` class we must first implement the `rollout_uniform` helper function which performs a rollout from a given game state. Here are some useful Game API methods/attributes that might be useful:

- `game.get_valid_moves()`
- `game.make_move(action_index)`
- `game.done`
- `game.get_reward(player_index)`

For now don't worry about the API implementation, you can assume the functions do what you think they do.

Hint: You might want to make use of `np.random.choice`

```

1 def rollout_uniform(game, rollout_player=None):
2     if rollout_player is None:
3         rollout_player = game.current_player
4
5     while ..... :
6         actions = ..... # How can we get actions?
7         a = ..... # How can we pick an action?
8
9         ..... # What else is there left to do?
10
11     return ..... # What should we return?
```

Based on the implementation above answer these questions:

1. If players alternate turns, whose perspective is the returned reward from?
2. Why can two rollouts from the same state produce different values?
3. In AlphaZero, what component replaces rollouts?

1.b FlatMCSearch

Now, let's implement `__init__` and `search` of `FlatMCSearch`.

Hint: You might want to make use of `np.random.choice` and other `np` functions. Also, as a reminder, the softmax function is as follows:

$$\sigma(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}}$$

```

1 class FlatMCSearch:
2     def __init__(self, num_simulations=100):
3
4         ..... # What should we initialize?
5
6     def search(self, game):
7         # Don't mutate this game! Make a copy with game.clone() instead
8         actions = game.get_valid_moves()
9         root_player = game.current_player
10
11         N_a = defaultdict(int) # What does this store?
12         sum_g_a = defaultdict(float) # What does this store?
13
14         for _ in range(.....): # What do we want to loop over?
15             game_copy = game.clone()
16
17             a = ..... # Pick a random action
18             N_a[a] += 1
19
20             game_copy.make_move(a)
21
22             # Since we already did game_copy.make_move(a), the current player is
23             # now root player's opponent, so we must pass in root_player
24
25             g = rollout_uniform(game_copy, root_player) # What does g stand for?
26             sum_g_a[a] += g # What does sum_g_a store?
27
28             # Why do we initialize policy to a vector of -infinity?
29             policy = float('-inf') * np.ones(game.get_action_size(), dtype=np.float32)
30
31             for a in N_a:
32                 policy[a] = ..... # Average score of each action
33
34             # Softmax
35             numer = .....
36
37             denom = .....
38
39             policy = ..... / .....
40
41         return policy

```

(Select all that apply) Which MCTS phases are implemented in the code above?

- ☐ Selection
- ☐ Expansion
- ☐ Rollout/Simulation
- ☐ Backup

2 Approximate Q-Learning

You decide to go to Kennywood this weekend. Our RL problem is based on choosing a ride from a set of many rides.

You start off feeling well, getting positive rewards from rides, some larger than others. However, there is some chance of each ride making you sick. If you continue going on rides while sick there is some chance of becoming well again, but you don't enjoy the rides as much, receiving lower rewards (possibly negative).

You have never been to an amusement park before, so you don't know how much reward you will get from each ride (while well or sick). You also don't know how likely you are to get sick on each ride, or how likely you are to become well again. What you do know about the rides is:

Actions / Rides	Type	Wait	Speed
Steel Curtain	Rollercoaster	Long	Fast
Lil' Phantom	Rollercoaster	Short	Slow
Cranky's Tower	Drop Tower	Short	Fast
Pirate	Pendulum	Short	Slow
Leave the Park	Leave	Short	Slow

We will formulate this as an RL problem with two states, **well** and **sick**. Each ride corresponds to an action. The 'Leave the Park' action ends the current run through the problem. Taking a ride will lead back to the same state with some probability or take you to the other state. We will use a feature-based approximation to the Q-values, defined by the following four features and associated weights:

Features	Initial Weights
$f_0 = f_{\text{sick}}(s, a) = \begin{cases} 0, s = \text{Well} \\ -5, s = \text{Sick} \end{cases}$	$w_0 = 1$
$f_1 = f_{\text{type}}(s, a) = \begin{cases} 1, \text{ if } \text{action type is Rollercoaster} \\ 0, \text{ otherwise} \end{cases}$	$w_1 = 2$
$f_2 = f_{\text{wait}}(s, a) = \begin{cases} 1, \text{ if } \text{action wait is Short} \\ 0, \text{ otherwise} \end{cases}$	$w_2 = 1$
$f_3 = f_{\text{speed}}(s, a) = \begin{cases} 1, \text{ if } \text{action speed is Fast} \\ 0, \text{ otherwise} \end{cases}$	$w_3 = 0.5$

1. Calculate Q-values for each action, given the state is 'Sick'.
2. Apply a Q-learning update based on the sample ('Well', 'Steel Curtain', 'Sick', -10.5), using a learning rate of $\alpha = 0.5$ and discount of $\gamma = 0.5$. What are the new weights?
3. Now we will consider the exploration exploitation tradeoff in this amusement park. Assume we have the initial weights from the table above. What action will an ϵ -greedy approach choose from the "Well" state? If multiple actions could be chosen, give each action and its probability.

4. When running Q-learning another approach to dealing with this tradeoff is using an exploration function:

$$f(u, n) = u + \frac{k}{n}$$

- (a) How is this function used in the Q-learning equations?
- (b) What does u represent in the exploration function?
- (c) What does n represent in the exploration function?
- (d) What does k represent in the exploration function?