CMU DATA SCIENCE CLUB X CMU POKER CLUB PRESENTS...

CMU POKER AI COMPETITION

Welcome to the Poker AI Workshop!

Poker Terminology

A hand: Cards dealt to a player (hidden information)

A board: community cards that are dealt face up. In no limit holdem, these consist of a flop (3 cards), turn (1 card), river (1 card).

SB/BB: Money put in by both players originally in each round (1 and 2). These alternate between rounds.

Showdown: Once no more community cards should be dealt, and all betting action has completed, the player with the better hand wins the pot.

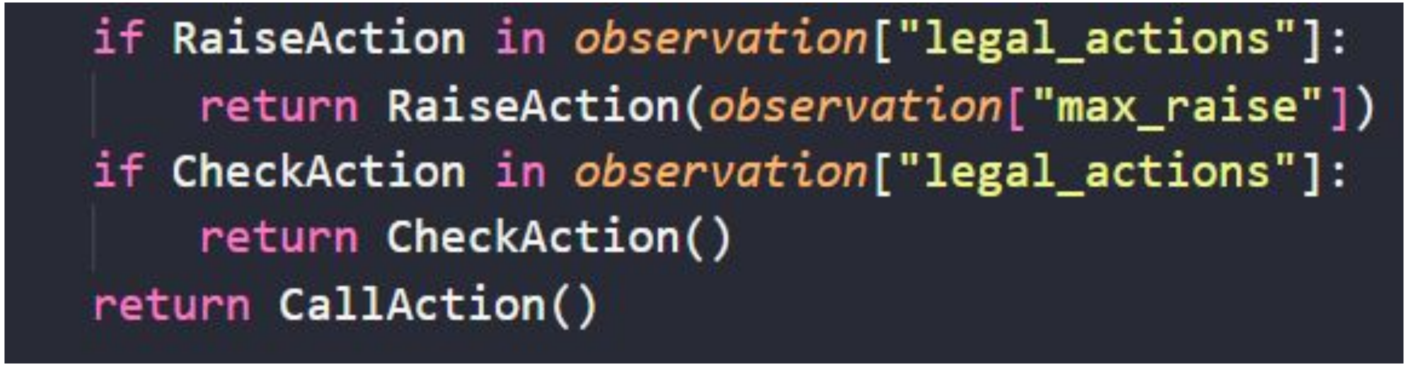
Check: A bet of 0 (passing)

Bet X: Bet X chips. Your opponent can either **call** the X, **raise** to **Y**, or **fold**.

All In: A bet or raise that wagers all of your remaining chips.

Contents

All-In Bot



```
if RaiseAction in observation["legal_actions"]:
    return RaiseAction(observation["max_raise"])
if CheckAction in observation["legal_actions"]:
    return CheckAction()
return CallAction()
```

A benchmark: Optimized All-In Bot

What if you only go all in with the *best* X% of hands and fold the rest? Or, what if instead of going all-in, you make a big preflop raise instead?

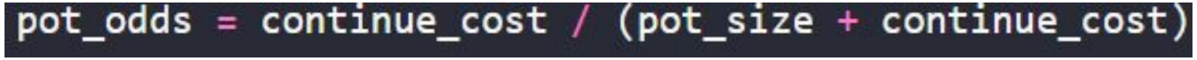
This bot can win money versus bots that call too wide. It can also win money versus bots that call too narrowly.

A *VERY FLAWED* model ("Prob Bot")

1. Calculate **Equity** (showdown winning probability)

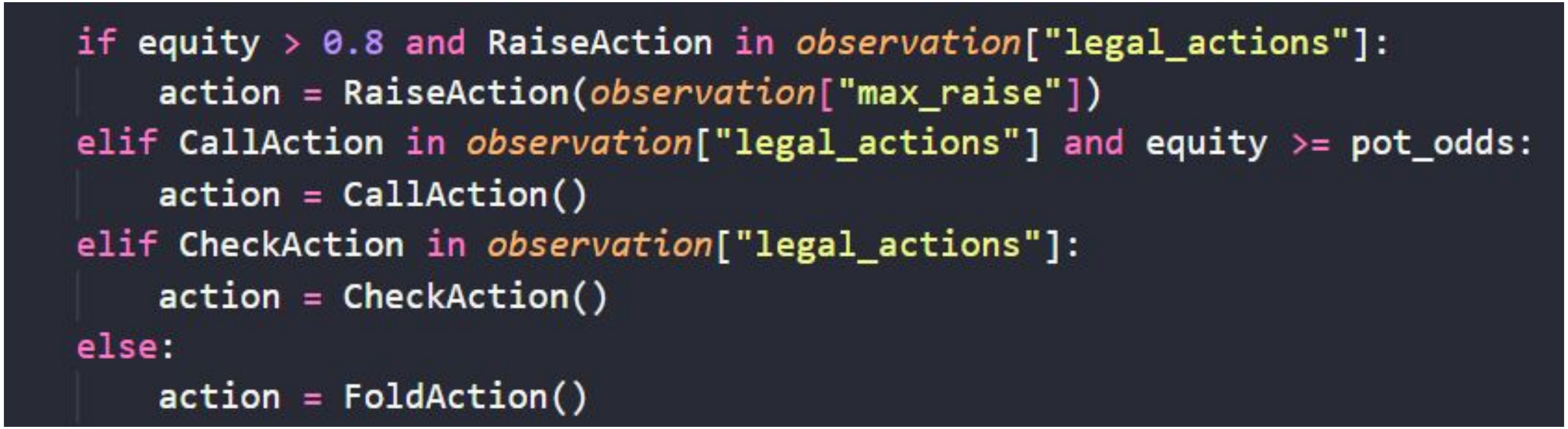
   Consider all possible combinations of enemy cards, community cards to count the number of times we win. Equity = win_num / total_num

2. Calculate **pot odds**



```
pot_odds = continue_cost / (pot_size + continue_cost)
```

3. Use the calculations



```
if equity > 0.8 and RaiseAction in observation["legal_actions"]:
    action = RaiseAction(observation["max_raise"])
elif CallAction in observation["legal_actions"] and equity >= pot_odds:
    action = CallAction()
elif CheckAction in observation["legal_actions"]:
    action = CheckAction()
else:
    action = FoldAction()
```

prob_bot Bankroll: 31605
allin_bot Bankroll: -31605

Why is this model *FLAWED*?

1. Over-Simplistic Equity Calculation (No attempts to narrow down hand range)
2. Inflexible Betting Amounts
3. Ignoring Contextual Information (Bet history, etc)
4. Lack of Bluffing Strategy

“In conclusion, while the given pseudocode is a basic representation of decision-making in poker, it lacks the sophistication and adaptability required for a robust poker-playing algorithm.”

Utilizing Reinforcement Learning

1. CFR (Counterfactual Regret Minimization) based methods

2. Policy Gradients

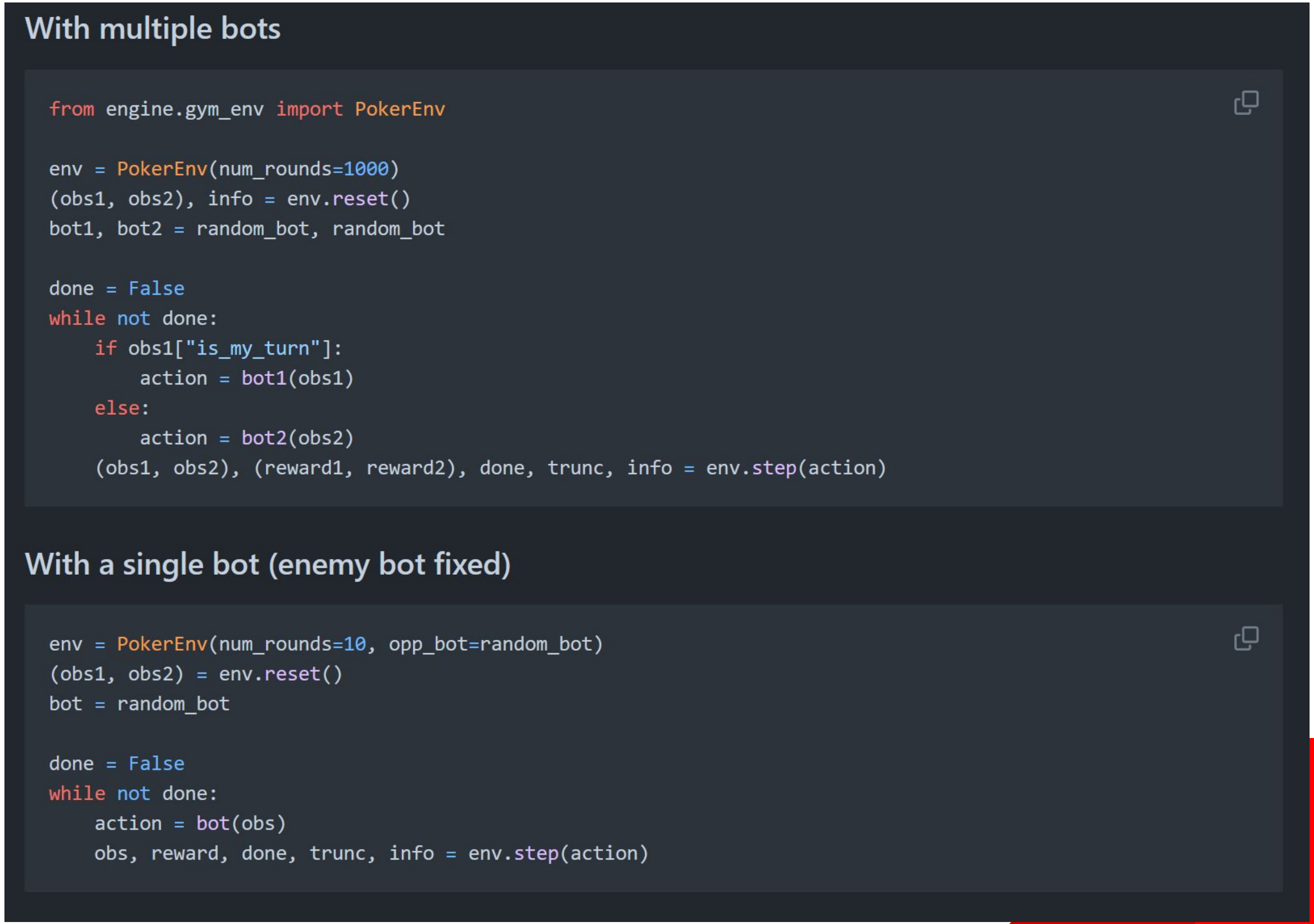
3. Genetic Algorithms

And much more…..

Utilizing Heuristic Algorithms (improving Prob Bot)

1. Evaluate your hand’s *Equity*: the expected share of the pot your hand is worth. Multiple factors influence a hands equity:
    a. Its current raw strength
    b. Its ability to improve to a stronger hand on certain cards (i.e. flush draws or straight draws in NLH).
    c. Its “blocking effect”: in making very strong hands in the villains range less likely.
    d. The position (the player who is last to act is mathematically known to have an advantage in almost all imperfect information games).
2. When to bet?
    a. For “value”: your hand has enough equity to want to inflate the size of the pot.
    b. As a “bluff”: your hand will almost certainly lose if you give up and check.

And much more…..



With multiple bots

```python
from engine.gym_env import PokerEnv

env = PokerEnv(num_rounds=1000)
(obs1, obs2), info = env.reset()
bot1, bot2 = random_bot, random_bot

done = False
while not done:
    if obs1["is_my_turn"]:
        action = bot1(obs1)
    else:
        action = bot2(obs2)
    (obs1, obs2), (reward1, reward2), done, trunc, info = env.step(action)
```

With a single bot (enemy bot fixed)

```python
env = PokerEnv(num_rounds=10, opp_bot=random_bot)
(obs1, obs2) = env.reset()
bot = random_bot

done = False
while not done:
    action = bot(obs)
    obs, reward, done, trunc, info = env.step(action)
```

GLHF!