Artificial Intelligence Project Report

By

Abhishek Pulicherla(Swedish ID No: 199501245857)
Jayanthan Subbaiah(Swedish ID No:199405243917)

Aim:

In this project, we designed a poker playing agent which plays according to the hand it has currently.

First, the agent had to analyze hands based on the current hands. In Poker, there are eight hands. The hands in the order of their strengths are as follows

1) Straight Flush
2) Four of a Kind
3) Full House
4) Flush
5) Straight
6) Three of a Kind
7) Two Pair
8) One Pair
9) High Cards

For this hand evaluation, we wrote a function **def handstrength(hand).** In this function, we converted the cards into a list to access the suits and the numbers of the hand.

```
def handstrength(hand):
    print(hand)
    cards= ''.join(hand)
    card_list = list(cards)
    #print(card_list)
    suits = card_list[1::2]
    numbers = card_list[0::2]
    #print(numbers)
    for i, num in enumerate(numbers):
        if num == "A":
            numbers[i] = 14
        elif num == "K":
            numbers[i] = 13
        elif num == "Q":
            numbers[i] = 12
        elif num == "J":
            numbers[i] = 11
        elif num == "T":
            numbers[i] = 10
    print(numbers)
    number_dict = collections.defaultdict(int)
    suit_dict = collections.defaultdict(int)
```

Then, we used defaultdict and assigned it to two variables number_dict and suit_dict. We used this to get the common cards and determine hands such as four of a kind, full house, three of a kind. We then used these variables to analyze hands

```
# 4-of-a-kind
if len(number_dict) == 2:
    if 4 in number_dict.values():
        rank = Hands[1]
        return rank
        print(rank)
    # Full house

elif len(number_dict) == 2:
    if 3 in number_dict.values():
        for my_card in numbers:
            if my_card == "K":
                count = count + 1
        if count == 2:
            rank = Hands[3]
            return rank
            print(rank)

    # flush

elif len(suit_dict) == 1:
    if 5 in suit_dict.values():
        rank = Hands[4]
        return rank
        print(rank)

    # straight

elif len(number_dict) == 5:

    max_val = (max([numbers.index(x) for x in number_dict.keys()]))
    min_val = (min([numbers.index(x) for x in number_dict.keys()]))
    # print(max_val,min_val)
    if int(max_val) - int(min_val) == 4:
        rank = Hands[5]
        return rank
        print(rank)


    # 3-of-a-kind
elif len(number_dict) == 3:
    if 3 in number_dict.values():
        rank = Hands[6]
        return rank
        print(rank)
```

In the function itself we have ranked the hands by declaring **rank=Hands[].** Then we used these ranks to choose our action, i.e. whether to check, fold or All in.

```
def queryOpenAction(_minimumPotAfterOpen, _playersCurrentBet,
_playersRemainingChips):
    print("Player requested to choose an opening action.")

    # Random Open Action
    def chooseOpenOrCheck():

        if _playersCurrentBet + _playersRemainingChips > _minimumPotAfterOpen:
            #return ClientBase.BettingAnswer.ACTION_OPEN,  iOpenBet
            return ClientBase.BettingAnswer.ACTION_OPEN,  (random.randint(0, 10)+
```

```
_minimumPotAfterOpen) if _playersCurrentBet + _playersRemainingChips +
random.randint(0, 10)> _minimumPotAfterOpen else _minimumPotAfterOpen
        else:
            return ClientBase.BettingAnswer.ACTION_CHECK

    handstrength(hand)
    if (rank==Hands[1] or rank==Hands[2]):

        return {
            0: ClientBase.BettingAnswer.ACTION_CHECK,
            1: ClientBase.BettingAnswer.ACTION_ALLIN,
        }.get(1, chooseOpenOrCheck())
    else:
        return {
            0: ClientBase.BettingAnswer.ACTION_CHECK,
            1: ClientBase.BettingAnswer.ACTION_ALLIN,
        }.get(random.randint(0,1), chooseOpenOrCheck())
```

Workflow of the Program: