Blackjack with Monte-Carlo ES

```
import numpy as np
import random
import itertools

# blackjack

# observation (=state):
# triple ( integer, integer, integer )
# 1. integer: the player's score (12 ~ 21)
# 2. integer: the dealer's card score of upside (1 ~ 10)
# 3. integer: 1 if the player has at least an ace, and 0 otherwise

# action
# 0: hit
# 1: stay
# doesn't allow double down, surrender and split

# step types
STEPTYPE_FIRST = 0
STEPTYPE_MID = 1
STEPTYPE_LAST = 2

cardset = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10 ]
deck = None

def shuffle_deck():
    global deck
    # card deck (we don't care the suite, but, for gui game in future) - 3 sets
    deck = \
      list(itertools.product(range(4), cardset)) \
      + list(itertools.product(range(4), cardset)) \
      + list(itertools.product(range(4), cardset))

    random.shuffle(deck)

shuffle_deck()
```

```
# environment parameters
dealer = None # dealer's hands
player = None # player's hands

# reset the environment
def generate_start_step():
    global dealer, player

    shuffle_deck()

    dealer = [ deck.pop(), deck.pop() ]
    player = [ deck.pop(), deck.pop() ]

    dealer_score = dealer[0][1]

    if player[0][1] == 1 and player[1][1] == 1:
        # if player gets double ace, the second one is counted as 1
        player_score = 12
        has_ace = 1
    elif player[0][1] == 1:
        player_score = 11 + player[1][1]
        has_ace = 1
    elif player[1][1] == 1:
        player_score = 11 + player[0][1]
        has_ace = 1
    else:
        player_score = player[0][1] + player[1][1]
        while player_score < 12:
            player.append(deck.pop())
            player_score += player[-1][1]
```

Blackjack with Monte-Carlo ES

```
        has_ace = 0

    # 1st step
    return { 'observation': (player_score, dealer_score, has_ace),
             'reward': 0., 'step_type': STEPTYPE_FIRST }
```

```
# returns a step, which is a dictionary { 'observation', 'reward', 'step_type' }
def generate_next_step(step, action):
    global player, dealer

    player_score, dealer_open, has_ace = step['observation']
    # has_ace is used to check if the player has
    # the option to count an ace as 1

    game_stop = False
    busted = False

    # with hit, get a card
    if action == 0:
        # hit - retrieve an additional card
        player.append(deck.pop())

        # note that an additional ace should be counted as 1
        player_score += player[-1][1]

        # if blackjack or bust, the game stops
        if player_score == 21:
            game_stop = True
        elif player_score > 21:
            # if busted but has an ace, the ace is counted as 1
            # and has_ace becomes false since already used
            if has_ace == 1:
                player_score -= 10
                has_ace = 0
            else:
                game_stop = True
                busted = True

    # with stay, game_stop
    else:
        game_stop = True

    # if busted, immediately the player loses
    if busted:
        return { 'observation': (player_score, dealer_open, has_ace),
                 'reward': -1., 'step_type': STEPTYPE_LAST }

    # now, if game_stop, it's dealer's turn & game stop
    if game_stop:
        dealer_has_ace = False
        dealer_busted = False

        # examine dealer's hands
        if dealer[0][1] == 1 and dealer[1][1] == 1:
            dealer_score = 12.
            dealer_has_ace = True
        elif dealer[0][1] == 1:
            dealer_score = 11. + dealer[1][1]
            dealer_has_ace = True
        elif dealer[1][1] == 1:
            dealer_score = 11. + dealer[0][1]
            dealer_has_ace = True
        else:
            dealer_score = dealer[0][1] + dealer[1][1]
            dealer_has_ace = False

        # the dealer takes cards until the score is at least 17
        while dealer_score < 17:
            dealer.append(deck.pop())
            dealer_score += dealer[-1][1]

            # if busted but has an ace, the ace is counted as 1
            if dealer_score > 21:
                if dealer_has_ace:
                    dealer_score -= 10
```

Blackjack with Monte-Carlo ES

```
                    dealer_has_ace = False
                else:
                    dealer_busted = True

        # compute the reward
        if dealer_busted:
            reward = 1.
        else:
            if player_score > dealer_score:
                reward = 1.
            elif player_score < dealer_score:
                reward = -1.
            else:
                reward = 0.

        return { 'observation': (player_score, dealer_score, has_ace),
                 'reward': reward, 'step_type': STEPTYPE_LAST }

    # continue
    else:
        return { 'observation': (player_score, dealer_open, has_ace),
                 'reward': 0., 'step_type': STEPTYPE_MID }
```

```
step = generate_start_step()
```

```
step
```

```
{'observation': (17, 10, 0), 'reward': 0.0, 'step_type': 0}
```

```
player
```

```
[(1, 9), (3, 8)]
```

```
dealer
```

```
[(3, 10), (3, 10)]
```

```
action = get_eps_soft_action(step)
```

```
action
```

```
1
```

```
step = generate_next_step(step, action)
```

```
step
```

```
{'observation': (16, 20, 0), 'reward': -1.0, 'step_type': 2}
```

```
dealer
```

```
[(3, 10), (3, 10)]
```

Blackjack with Monte-Carlo ES

```
player
```

```
[(1, 9), (3, 8)]
```