Assignment 1

by: Sharon Hendy - 209467158, Pan Eyal - 208722058

In this jupyter notebook we only show the results of the algorithms we wrote in the Blackjack.py file.
We represent states as tuples of (player sum, dealer showing, usable ace).
We added three special states that will be used as terminal states: WIN, LOSE, DRAW.
First we will find the transition matrix and the reward function, then with the policy iteration algorithm, we will find the optimal policy.
The policy iteration uses an initial policy that always hit if the player sum is less than 20 as specified in the assignment.
Finally, we will run and plot the value averages changes for each iteration during learning and plot the starting policies for unusable ace case.

In [1]:
```
from Blackjack import transition_matrix, reward_function, policy_iteration, ALL_STA

import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from matplotlib.patches import Patch
```

Get transition matrix and reward function

The transition matrix function calculates the transition matrix for the blackjack game.
The function gets a sample size and runs sample size blackjack games.
If the sampled action results in termination, the transitioned state will be determined by the received reward (WIN, LOSE, DRAW).
The function prints all non-zero transition probabilities with a 3 decimal accuracy and returns the transition matrix.
For the sake of keeping the PDF clean, we will not print the transition matrix here.

In [2]:
```
tr_matrix = transition_matrix(sample_size=10 ** 6, should_print=False)
```

```
100%|██████████| 1000000/1000000 [01:44<00:00, 9539.12it/s]
```

The reward function returns the already known reward function for the blackjack game that is:
-> for a terminating state s: R(s) = 1 for win, R(s) = -1 for lose, R(s) = 0 for draw.
-> for a non-terminating state s: R(s) = 0.

In [3]:
```
r_s_a = reward_function()
```

Create a specific policy that always hit if the player sum is less than 20

In [4]:
```
pi = dict()
for state in ALL_STATES:
    pi[state] = 1 if state[0] < 21 else 0
for state in [WIN, LOSE, DRAW]:
    pi[state] = 0
```

Run policy iteration

The policy iteration function gets a transition matrix, a reward function, a number of iterations and an initial policy.
The function iterates between the approximate policy evaluation and the policy improvement functions until the policy is stable.
The function returns the optimal value function, the optimal policy and the average value over iterations for plotting the desired graph.

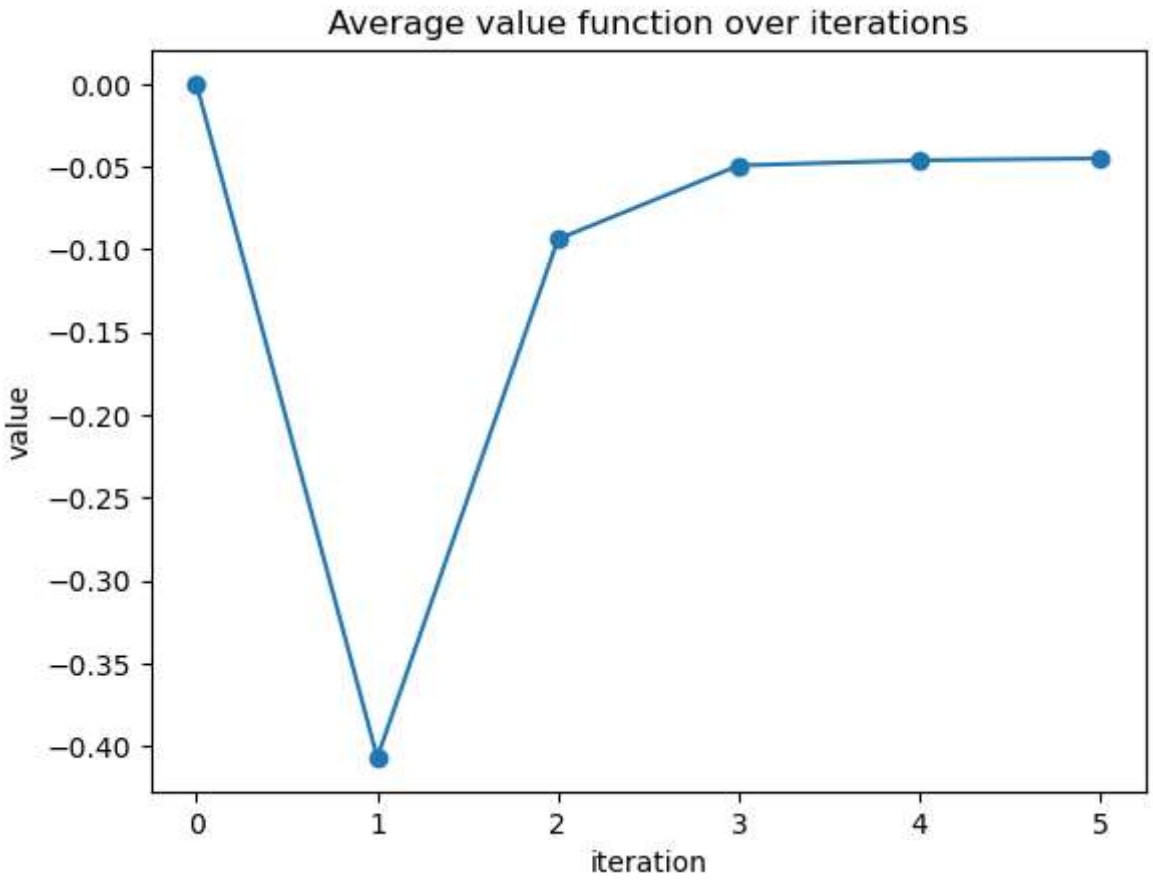
In [5]:
```
v, pi, value_avg = policy_iteration(tr_matrix, r_s_a, k=5, pi=pi)
```

```
Policy iteration converged after 5 iterations.
```

Plot value averages changes

Here we plot the average value over iterations.
We can see that during the k=5 run, the value average increases with each iteration (excluding the first iteration with initial values).

In [6]:
```
plt.plot(np.arange(len(value_avg)), value_avg, 'o-')
plt.title("Average value function over iterations")
plt.xlabel("iteration")
plt.ylabel("value")
plt.show()
plt.clf()
```




```
<Figure size 640x480 with 0 Axes>
```

Plot the starting policies for unusable ace

Here we plot the starting policies for unusable ace.

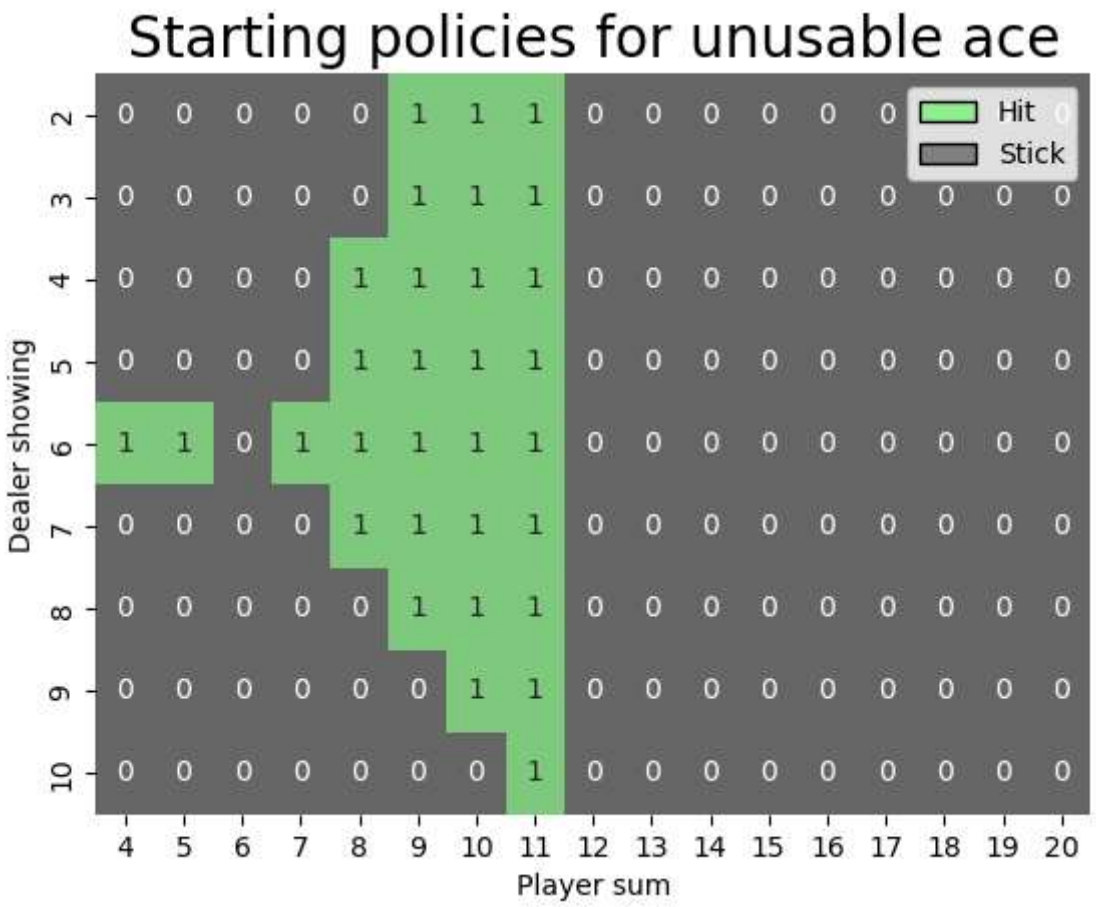
In [7]:
```
player_count, dealer_count = np.meshgrid(
    # players count, dealers face-up card
    np.arange(4, 21),
    np.arange(2, 11),
)

# create the policy grid for plotting
pi_grid = np.apply_along_axis(
    lambda obs: pi[(obs[0], obs[1], 0)],
    axis=2,
    arr=np.dstack([player_count, dealer_count]),
)

# plot the policy for unusable ace
plt.figure()
sns.heatmap(pi_grid, linewidth=0, annot=True, cmap="Accent_r", cbar=False, xticklab
plt.title("Starting policies for unusable ace", fontsize=20)
plt.xlabel("Player sum")
plt.ylabel("Dealer showing")

# add a legend
legend_elements = [
```

```
    Patch(facecolor="lightgreen", edgecolor="black", label="Hit"),
    Patch(facecolor="grey", edgecolor="black", label="Stick"),
]
plt.legend(handles=legend_elements)
plt.show()
plt.clf()
```



<Figure size 640x480 with 0 Axes>