מבוא בבינה מלאכותית

דו"ח תרגיל בית –

אלגוריתמי למידה – mini poker

מגישים :

עאישה עואודה

316065200

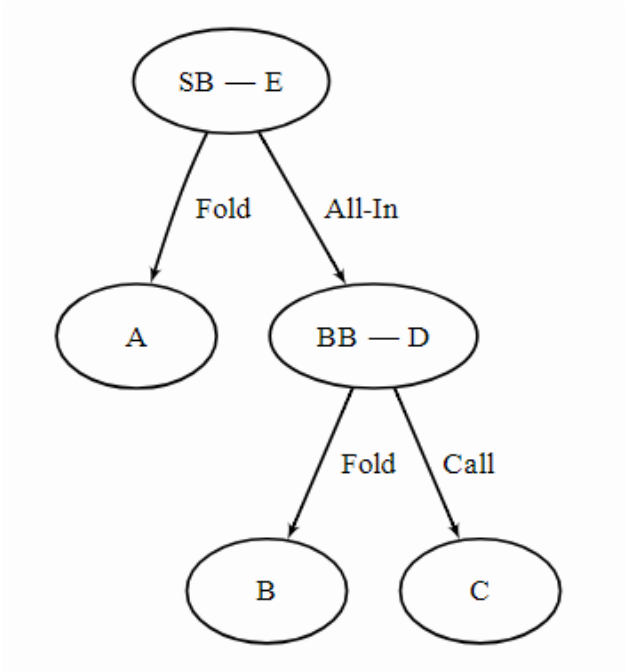
חסן כבהה

318242914

מיני פוקר (mini poker):

1. שני השחקנים מתחילים עם ערימות של S וביד 2 כרטיס באופן אקראי.
2. שחקן BB מהמר על 1.0 ציפ, ואת שחקן SB על 0.5 .
3. SB יכול ללכת all-in או fold.
4. ממול BB יכול לעשות All-in או fold.

בעיה כזאת אפשר ליצג אותה דרך עץ החלטה :



בדוח מציוג פרתון למשחק הזה דרך reinforcement learning עם משוואת q-learning.
המשוואה :

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

מ

<u>מידלנו את מרחב המשחק ל states ופעלנו לפי האלגוריתם של reinforcement learning</u> :

1. We are in some **state** (i.e. the state of the world, which we observe).
2. We use that info to take some **action**.
3. We get some **reward**.
4. Repeat.

אנו עושים זאת שוב ושוב: צופים בstate, בוחרים פעולה, מקבלים reward, צופים state חדש, לוקחים פעולה נוספת, מקבלים reward נוסף וכו '. הבעיה של RL היא פשוט להבין כיצד לבחור בפעולות כדי לקבל reward ככל האפשר .

1. Reward :

מכיוון שהסוכן הוא בעל מוטיבציה ל reward והוא ילמד כיצד לשלוט במשחק על ידי חוויות שקרו קודם בסביבה, עלינו להחליט על הגמול ו / או העונשים ועל גודלם בהתאם. הנה כמה נקודות לשיקול:

- הסוכן צריך לקבל reward חיובי גבוה עבור משחקון מוצלח כי התנהגות זו היא מאוד הרצוי
- הסוכן צריך להיענש אם הוא מפסיד במשחקון מסויים, כדי שלא יחזור על פעולה כזאת בעתיד.

לכן בחרנו את ה reward להיות : reward = rewardAfter – rewardBefore

ז"א ההפרש בין כמות הכסף בין כמה שיש לו ובין כמה שהיה לו, בגלל אם כמות הכסף אחרי המשחקון עלתה אז ה reward יהיה חיובי אחרת, זאת אומרת שהפסיד לכן הערך יהיה שלילי.

בקוד :

```
next_indexStateA = getIndex(next_stateA)
next_indexStateB =getIndex(next_stateB)
next_maxA = np.max(Qtable[next_indexStateA])
next_maxB = np.max(Qtable[next_indexStateB])
rewardA = balanceA - oldBalanceA
rewardB = balanceB - oldBalanceB
new_valueA = (1 - alpha) * old_valueA + alpha * (rewardA+ gamma * next_maxA)
new_valueB = (1 - alpha) * old_valueB + alpha * (rewardB + gamma * next_maxB)
```

2. State space :

ב reinforcement learning הסוכן נתקל ב state, ולאחר מכן פועל בהתאם לstate בה הוא נמצא.

State space -הוא מערכת של כל המצבים האפשריים שהשחקן שלנו יכול שנמצא בהן. state צריכה להכיל מידע שימושי הסוכן צריך לעשות את הפעולה הנכונה.

במשחק שלנו הגדרנו את ה state להכיל את ה hand שהסוכן שלנו מחזיק, ואם שני הכרטיסים שהוא מחזיק מאותו סוג ועוד שדה נוסף אם הסוכן שלנו משחק ראשון.

המטרה שלנו היא לצמצם במספר המצבים כדי שהיה הקוד יותר יעיל, לכן לא שמרנו את כל האפשריות של ה hand שזה $\binom{52}{2}$, אלא שמרנו רק את ערך הכרטיסים ואם הם מאותו סוג, בגלל זה לא משנה אם ה hand A♦A♣ או A♥A♣ בגלל שהסוג בשני המקרים שווה לכן הן אותו מצב. ובנוסף 6♣2♣ וגם 6♥2♥ הן אותו מצב ולכן נקבל שמספר המצבים הוא 4*13*13, כפול 4 בגלל אם הוא מתחיל ראשון ואם שני הכרטיסים אותו סוג.

State = {card1, card2, ifSameSuit, ifSb}

לכן בקוד השתמשתי ב list כדי לשמור את כל המצבים השונים.

```
counter=0
for i in range(2,15):
    for j in range(2,15):
        state = [i,j,0,0]
        statelist.append(state)


for i in range(2,15):
    for j in range(2,15):
        state = [i,j,1,0]
        statelist.append(state)

for i in range(2,15):
    for j in range(2,15):
        state = [i,j,0,1]
        statelist.append(state)

for i in range(2,15):
    for j in range(2,15):
        state = [i,j,1,1]
        statelist.append(state)

Qtable = np.zeros((len(statelist),2))

np.save('Qtable.npy', Qtable)
```

<u>הסבר :</u>

ב 4 לולאות for עברתי על כל הערכים האפשריים ל hand וב state[2] שומרים אם שני הכרטיסים מאותו סוג, וב state[3] שומר אם הסוכן שלנו מתחיל ראשון.

3. Action space :

הסוכן נתקל באחד מה states וזה לוקח פעולה. הפעולה במקרה שלנו יכולה להיות fold/ All-in .

במילים אחרות, יש לנו שני פעולות אפשריות:

- Fold
- All-in

4. Penalty :

בשלב הלימוד צריכים להגדיר מתי הסוכן מקבל עונש, ולמדוד את מס' העונשים שהסוכן קיבל עם האיטירציות של הלימוד. הגדרת ה penalty :

```
if rewardA < 0 :
    penalty = penalty+1
```

כאשר:

```
rewardA = balanceA - oldBalanceA
```

כלומר שבכל פעם שמשחקים משחקון נקבל שסכום הצ'פ ירד ממה שהיה לפני שנשחק (הסוכן הפסיד) אז המגדילים את penalty באחד.

Implementation with Python:

```
Qtable = np.zeros((len(statelist),2))

np.save('Qtable.npy', Qtable)
```

הגדרת את Q-table שנשמור בה את הערכים.

Qtable : בהתחלה מבני הנתונים הוא מאותחל עם אפסים, הוא בגודל statelist שהוא מייצגת את כל המצבים, ובמימד השני שם לנו את שתי הפעולות האפשריות fold ו All-in. בגלל שבכל state יכולים לעשות שתי פעולות fold OR All-in

מאגר הערכים Qtable הם מפתחים לשילוב (state,action).

הסבר על האלגוריתם:

בשלב האימון כדי שבאימון יהיה יותר יעיל החלטתי ששני הסוכנים ישתמשו בטלבה כדי שיקבעו את הפעולה הטובה ביותר שיקחו.

השימוש בטבלה:

```
while not done:


        if random.uniform(0, 1) < epsilon:    # Explore action space
            if random.uniform(0, 1) < 0.5:
             action1 = 0    # fold
            else:
             action1 =1     # all in
        else:

            action1 = np.argmax(Qtable[stateIndexA])   # Exploit learned values

        if random.uniform(0, 1) < epsilon:   # Explore action space
            if random.uniform(0, 1) < 0.5:
                action2 = 0   # fold
            else:
                action2 = 1   # all in
        else:

            action2 = np.argmax(Qtable[stateIndexB])   # Exploit learned values
```

אנחנו צריכים לוודא לקחת את כל הפעולות בכל המצבים לפחות מדי פעם אם אנחנו רוצים הערכות טובות. אז, יש על השחקנים לפעול באופן אקראי חלק קטן ε של הזמן אבל אחרת להשתמש בערכים של Qtable . לכן אם הערך האקראי גדול מ ε נשתמש בטבלה אחרת בוחרים את הפעולה באופן אקראי.

עדכון הטבלה:

```
#######################################################
balanceA, balanceB, sb = onePly(balanceA,balanceB,board,player1_hand,player2_hand,sb,actionA,actionB,evaluator)

#######################################################
if balanceB == 0 or balanceA == 0:
    done = 1
old_valueA = Qtable[stateIndexA, action1]
old_valueB = Qtable[stateIndexB, action2]
deck=Deck()
cardsNewRoundPlayer1 = deck.draw(2)
cardsNewRoundPlayer2 = deck.draw(2)
next_stateA= createState(cardsNewRoundPlayer1,sb)
next_stateB = createState(cardsNewRoundPlayer2, sb)
next_indexStateA = getIndex(next_stateA)
next_indexStateB =getIndex(next_stateB)
next_maxA = np.max(Qtable[next_indexStateA])
next_maxB = np.max(Qtable[next_indexStateB])
rewardA = balanceA - oldBalanceA
rewardB = balanceB - oldBalanceB
new_valueA = (1 - alpha) * old_valueA + alpha * (rewardA+ gamma * next_maxA)
new_valueB = (1 - alpha) * old_valueB + alpha * (rewardB + gamma * next_maxB)
Qtable[stateIndexA, action1] = new_valueA  # too slow
Qtable[stateIndexB, action2] = new_valueB
```

משחקים משחקון בין שני השחקנים דרך פונקצית onePly ומעדכינים את הערימות של השחקנים בהתאם. ואחר מכך מעדכינים את ה Qtable לפי הנוסחה שהצגנו קודם.

תוצאות:

1- אחוז הפעים שהסוכן החכם הצליח לבלף את יריבו :

בהתחלת התוכנית עשינו אימון דרך האלגוריתם של Q-learning ובדרך כזאת הצלחנו למלא את הטבלה Qtable כך הטבלה זאת תעזור לסוכן שלנו לקחת את הפעולה הטובה ביותר בתוך המשחק. עשינו את האימון כ 100,000 משחק שכל משחק מורכב מכמה משחקונים. ובכך דאגנו שהטבלה תתמלא בצורה הטובה ביותר. אחר האימון שיחקנו מול שחקן שמשחק באופן אקראי, כך ששיחקו 100,000 משחק חזרנו על הניסוי 10 פעמים כדי לוודא את התוצאות. והתוצאות הן :

תוצאה 1 :

```
Do you want to play? (y/n)n
The agent will play vs random player
number games our agent win :  60047

Process finished with exit code 0
```

תוצאה 2 :

```
Do you want to play? (y/n)n
The agent will play vs random player
number games our agent win :  59543

Process finished with exit code 0
```

תוצאה 3 :

תוצאה 4 :

```
Do you want to play? (y/n)n
The agent will play vs random player
number games our agent win :  60338

Process finished with exit code 0
```

תוצאה 5 :

```
Do you want to play? (y/n)n
The agent will play vs random player
number games our agent win :  59098

Process finished with exit code 0
```

תוצאה 6 :

```
Do you want to play? (y/n)n
The agent will play vs random player
number games our agent win :  59588

Process finished with exit code 0
```

תוצאה 7 :

```
Do you want to play? (y/n)n
The agent will play vs random player
number games our agent win :  59491

Process finished with exit code 0
```

תוצאה 8:

```
Do you want to play? (y/n)n
The agent will play vs random player
number games our agent win :  59574

Process finished with exit code 0
```

תוצאה 9 :

```
Do you want to play? (y/n)n
The agent will play vs random player
number games our agent win :  59176

Process finished with exit code 0
```

תוצאה 10 :

```
Do you want to play? (y/n)n
The agent will play vs random player
number games our agent win :  59659

Process finished with exit code 0
```

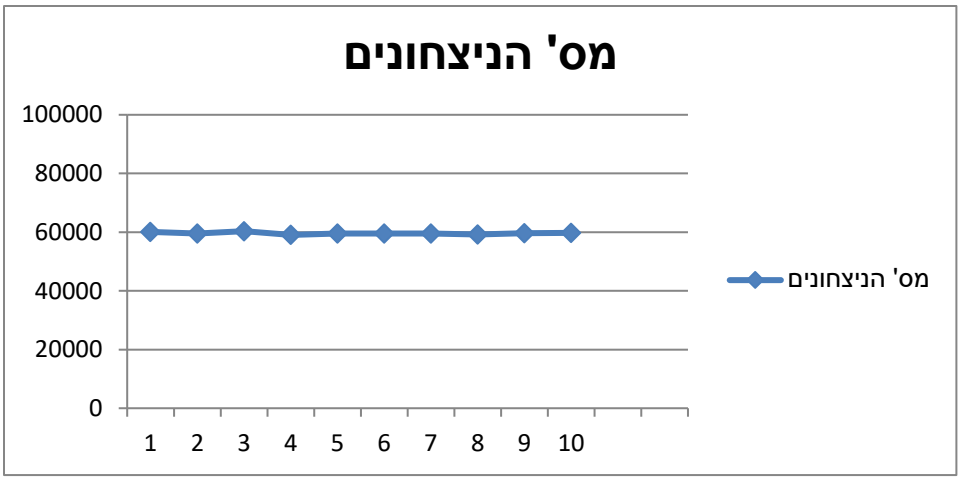
```
Do you want to play? (y/n)n
The agent will play vs random player
number games our agent win :  59744

Process finished with exit code 0
```

סיכום לתוצאות :

מספר הניסוי	תוצאה (מספר הניצחונים מ 100,000)
1	60047
2	59543
3	60338
4	59058
5	59588
6	59491
7	59574
8	59176

9	59659
10	59744

דרך גרף :



הסבר :

עשינו את הניסוי 10 פעמים ובל 10 הפעמים קיבלנו שהסוכן שלנו מנצח בקרוב 60000 מתוך 100000 המשחקים, כלומר אחוז הנצחון שקיבלנו הוא 60% מכל המשחקים, שזה אחוז יותר טוב מלשחק באופן אקראי.

תוצאות הנסוי אינן תוצאות אקראיות בגלל שחזרנט על הנסוי 10 פעמים ובכל 10 הפעמים קיבלנו ערכים קרובים.

כלומר הסוכן שלנו השתמש ב Qtable שחישבנו בצעד האימון, והטבלה נתנה לסוכן העדפה לנצח את השחקן האקראי ב 60% מהמשחקים. כלומר בכל state מסויים הסוכן חזר לטבלה ולפיה הוא ידע את הצעד הטוב לעשות.

2- ממוצע המשחקונים וממוצע העונשים הלומד קיבל:

תוצאה 1 :

```
Trainig :
Trainig finished
the penalty average:  3.55929
Do you want to play? (y/n)n
The agent will play vs random player
Playing:
number games our agent win :  60070
number rounds each game on average 6.37098
```

תוצאה 2 :

```
Trainig :
Trainig finished
the penalty average:  3.51632
Do you want to play? (y/n)n
The agent will play vs random player
Playing:
number games our agent win :  59545
number rounds each game on average 6.02999
```

תוצאה 3 :

```
Trainig :
Trainig finished
the penalty average:  3.57497
Do you want to play? (y/n)n
The agent will play vs random player
Playing:
number games our agent win :  59919
number rounds each game on average 6.44215
```

תוצאה 4 :

```
Trainig :
Trainig finished
the penalty average:  3.55295
Do you want to play? (y/n)n
The agent will play vs random player
Playing:
number games our agent win :  60004
number rounds each game on average 6.61989
```

תוצאה 5 :

```
Trainig :
Trainig finished
the penalty average:  3.61018
Do you want to play? (y/n)n
The agent will play vs random player
Playing:
number games our agent win :  59832
number rounds each game on average 6.23268
```

תוצאה 6:

```
Trainig :
Trainig finished
the penalty average:  3.66732
Do you want to play? (y/n)n
The agent will play vs random player
Playing:
number games our agent win :  59729
number rounds each game on average 6.35547
```

תוצאה 7:

```
Trainig :
Trainig finished
the penalty average:  3.54789
Do you want to play? (y/n)n
The agent will play vs random player
Playing:
number games our agent win :  59727
number rounds each game on average 6.30085
```

תוצאה 8:

```
Trainig :
Trainig finished
the penalty average:  3.54381
Do you want to play? (y/n)n
The agent will play vs random player
Playing:
number games our agent win :  58963
number rounds each game on average 5.87617
```

הסבר התוצאות :

בשלב האימון קיבלנו שממוצע העונשים שהסוכן שלנו קיבל הוא 3.55 , חזרנו על האימון 10 פעמים כדי לוודא את התוצאות וב 10 הפעמים קיבלנו ערכים מאוד קרובים. הסוכן קיבל עונש בכל פעם הוא טעה במשחקון, ז"א בכל פעם שקיבלנו ערך שלילי ל reward עקב ההפסד במשחקון.

מספר המשחקונים הממוצע שהסוכן שחק עד שנצח את יריבו הוא 6.2, באופן דומה חזרנו על הניסוי 10 פעמים לוודא את התוצאות, והערכים היו מאוד קרובים.

3- פרמטרי הלמידה שהשתמשנו בהן :

```
def train(evaluator):

    # Hyperparameters
    alpha = 0.01
    gamma = 0.3
    epsilon = 0.2
```

Qvalue מאותחלים לערך אפס וכאשר הסוכן חושף את עצמו לסביבה ומקבל reward שונים על ידי ביצוע פעולות שונות, ערכי Qtable מתעדכנים באמצעות המשוואה של ה Qlearning לכן לבחירת הערכים של הפרמטרים חשיבות גדולה :

- α (אלפא) הוא קצב הלמידה (0<α≤1) - α הוא המידה שבה ערכי Q שלנו מתעדכנים בכל איטרציה. לכן במצב שלנו בחרו α = 0.01 . הבחירה היתה מכמה סיבות והן :

- רוצים שערכים הטבלה יתעדכנו בקצב איטי בגלל שיש לנו 100,000 איטירציה, כך שכל איטירציה תתרום לערך הנוכחי בכמות קטנה של שינוי. בגלל אם באיטירציה מסויימת קיבלנו ערך גדול לא רוצים שהערך שנמצא בטבלה יושפע במידה גדולה.
- תמיד בכל האלגורתמים צריך שקצב הלמידה יהיה איטי כך שהאלגוריתם ילמד מכל האיטירציות, ובכך נקבל ערכים טובים ומשקפים את איכות ביצוע הפעולה כאשר נמצאים במצב מסוים באופן הטוב ביותר.

- γ הוא גורם ההפחתה (0<γ ≤1) - קובע כמה חשיבות אנחנו רוצים לתת לתגמולים עתידיים. ערך גבוה (קרוב ל 1) נותן חשיבות גדולה ל reward לטווח הארוך, ואם γ קרוב ל 0 גורם לסוכן שלנו לשקול רק תגמול מיידי, ומכאן מה שהופך אותו לחמדן (greedy).
במשחק שלנו בחרנו γ = 0.3 :

- אנחנו רוצים ל reward עתידיים יהיה להם השפעה אבל לא השפעה גדולה לכן בחרנו אותו 0.3 כך שמתחשבים ב reward עתידי אבל לא במידה גדולה . וה reward החשוב ביותר הוא הנוכחי שקיבלנו אותו מה state הנוכחי.

- ε : במשחק בחרנו את ε להיות 0.2, כלומר ב 0.2 מהמקרים עושים explore ולא להשתמש בטבלה כדי לנסות פתרונות שונים מהטבלה כדי שלא נתקע ב local optima ותמיד לנסות בפעולות חדשות, וב 0.8 מהפעמים להשתמש בערך הטוב ביותר שנמצא בטבלה. אנחנו רוצים שברוב הפעמים לשתמש בטבלה(0.8) אבל לא בכל מידי פעם לנסות לעשות explore (0.2) .

4- **גרף שיפור הביצועים של הסוכן החכם לאורך מספר הנסיונות**:

תוצאות (100,000 משחק):

10 iteration :

```
Number iterations in training : 10
Trainig finished
the penalty average:  6e-05
Do you want to play? (y/n)n
The agent will play vs random player
Playing:
number games our agent win :  50085
number rounds each game on average 2.10963

Process finished with exit code 0
```

50 iteration :

```
Number iterations in training : 50
Trainig finished
the penalty average:  0.00028
Do you want to play? (y/n)n
The agent will play vs random player
Playing:
number games our agent win :  49807
number rounds each game on average 2.30103

Process finished with exit code 0
```

100 iteration :

```
Number iterations in training : 100
Trainig finished
the penalty average:  0.00075
Do you want to play? (y/n)n
The agent will play vs random player
Playing:
number games our agent win :  50431
number rounds each game on average 2.55137
```

500 iteration :

```
Number iterations in training : 500
Trainig finished
the penalty average:  0.0071
Do you want to play? (y/n)n
The agent will play vs random player
Playing:
number games our agent win :  51915
number rounds each game on average 4.24627
```

1000 iteration:

```
Number iterations in training : 1000
Trainig finished
the penalty average:  0.01983
Do you want to play? (y/n)n
The agent will play vs random player
Playing:
number games our agent win :  53351
number rounds each game on average 5.0384

Process finished with exit code 0
```

5000 iteration:

```
Number iterations in training : 5000
Trainig finished
the penalty average:  0.12723
Do you want to play? (y/n)n
The agent will play vs random player
Playing:
number games our agent win :  55511
number rounds each game on average 4.65398

Process finished with exit code 0
```

10,000 iteration:

```
Trainig :
Number iterations in training : 10000
Trainig finished
Do you want to play? (y/n)n
The agent will play vs random player
Playing:
number games our agent win :  56258
number rounds each game on average 4.60559

Process finished with exit code 0
```

50,000 iteration:

```
Trainig :
Number iterations in training : 50000
Trainig finished
Do you want to play? (y/n)n
The agent will play vs random player
Playing:
number games our agent win :  59839
number rounds each game on average 6.06888

Process finished with exit code 0
```

100,000 iteration:

```
Trainig :
Number iterations in training : 100000
Trainig finished
Do you want to play? (y/n)n
The agent will play vs random player
Playing:
number games our agent win :  59632
number rounds each game on average 6.34487

Process finished with exit code 0
```

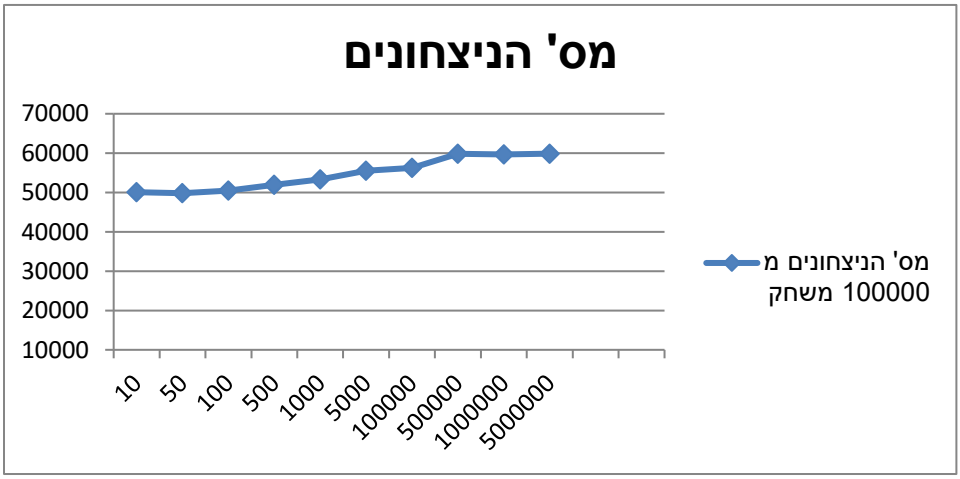
500,000 iteration:

```
Trainig :
Number iterations in training : 500000
Trainig finished
Do you want to play? (y/n)n
The agent will play vs random player
Playing:
number games our agent win :  59858
number rounds each game on average 5.9865

Process finished with exit code 0
```

דרך טבלה:

מס' איטירציות באימון	מס' הניצחונים מ 100000 משחק
10	50085
50	49807
100	50431
500	51915
1000	53351
5000	55511
100000	56258
500000	59839
1000000	59632
5000000	59858

דרך גרף :



הסבר את התוצאות:

לפי התוצאות יכולים לראות ככל שאורך תקופת האימון גדלה (מספר האיטירציות בתקופת האימון) מספר הניצחונים של הסוכן גדלה על חד מסויים. אפשר להסביר את התוצאות האלו לפי :

- ככל שהסוכן מתאמן יותר אז הטבלה Qtable תתמלא בצורה יותר טובה, כך שנדאג שכל התאים באבלה התמלאו וגם על כל תא התעדכן מספר פעמים עם האימון כך שהערך הנוכחי של התא משקף בצורה קרובה ל Quality של התא הזה.
- אימון לטווח קטן (מספר איטירציות קטן) לא יעיל בגלל הערכים שבטבלה התקבלו ממספר קטן של נסיונות, ואם באחת הניסיונות קיבלנו ערך חריג (ערך גדול או קטן)

אז נקבל שהערך הנמצא לא משקף טוב, ולכן צריך לחזור על האימון מספר רב של פעמים כדי להתפטר השגיאות (סטיית התקן)

- השיפור במספר הניצחונים עם הגדלת אורך הנסיונות הוא עד גבול מסויים, בגלל שאחר חד כזה הטבלה תהיה מעודכנת בצורה הטובה ביותר ולכן המשכת האימון של הסוכן לא תתרום בהגדלת מספר הניצחונים.
- אי אפשר להתעלם מהעובדה שהניצחון תלוי בכריטים שירדו על השולחן ובסופו של דבר זה עניין של מזל, לכן אימון מיותר של הסוכן לא תגרום בתועלת למספר הניצחונים.
- אפשר לראות כאשר הטבלה לא מעודכנת בצורה טובה יש לסוכן שלנו אחוז לנצח בערך 50% שזה שקול לשחק באופן אקראי ללא טבלה. אבל כאשר הטבלה התעכנה בצורה טובה יותר רואים שהאוז הזה גדל ל 60% .