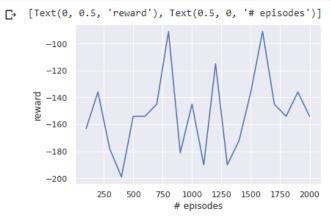
Q-Learning (ALPHA=0.1 EP=2000)

▼ And the rewards

```
x_points = np.arange(LOG_INTERVAL, NR_EPISODES + 1, LOG_INTERVAL)
y_points = REWARDS

plt = sns.lineplot(x=x_points, y=y_points)
plt.set(xlabel="# episodes", ylabel="reward")
```

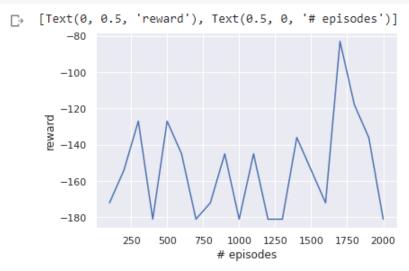


SARSA(ALPHA=0.1 EP=2000)

And the rewards

```
x_points = np.arange(LOG_INTERVAL, NR_EPISODES + 1, LOG_INTERVAL)
y_points = REWARDS

plt = sns.lineplot(x=x_points, y=y_points)
plt.set(xlabel="# episodes", ylabel="reward")
```



Q-Learning(ALPHA = 0.3 EP = 3000)

```
x_points = np.arange(LOG_INTERVAL, NR_EPISODES + 1, LOG_INTERVAL)
       y_points = REWARDS
        plt = sns.lineplot(x=x_points, y=y_points)
        plt.set(xlabel="# episodes", ylabel="reward")
   [Text(0, 0.5, 'reward'), Text(0.5, 0, '# episodes')]
            -80
           -100
           -120
        D −140

ma

−160
           -180
           -200
           -220
                      500
                             1000
                                    1500
                                           2000
                                                  2500
                                                         3000
                0
                                  # episodes
▼ 4.2 SARSA (25p)
```

SARSA(ALPHA = 0.3 EP = 3000)

```
x_points = np.arange(LOG_INTERVAL, NR_EPISODES + 1, LOG_INTERVAL)
    y_points = REWARDS
    plt = sns.lineplot(x=x_points, y=y_points)
    plt.set(xlabel="# episodes", ylabel="reward")
\rightarrow [Text(0, 0.5, 'reward'), Text(0.5, 0, '# episodes')]
       -100
       -120
       -140
       -160
       -180
             0
                   500
                         1000
                                1500
                                       2000
                                               2500
                                                      3000
                               # episodes
```

Q-LEARNING(ALPHA=0.3 GAMMA=0.5 EP = 3000)

▼ And the rewards

```
\begin{tabular}{lll} $x\_points = np.arange(LOG_INTERVAL), NR\_EPISODES + 1, LOG_INTERVAL) $y\_points = REWARDS \end{tabular}
     plt = sns.lineplot(x=x_points, y=y_points)
     plt.set(xlabel="# episodes", ylabel="reward")
[Text(0, 0.5, 'reward'), Text(0.5, 0, '# episodes')]
         -110
         -120
         -130
         -140
         -150
         -160
         -180
         -190
                      500
                             1000
                                     1500
                                              2000
                                                      2500
                                                             3000
                                    # episodes
```

SARSA(ALPHA=0.3 GAMMA=0.5 EP = 3000)

▼ And the rewards



Q-Learning(ALPHA=0.5 GAMMA=0.5 EP=5100)

▼ And the rewards

```
x_points = np.arange(LOG_INTERVAL, NR_EPISODES + 1, LOG_INTERVAL)
     y_points = REWARDS
     plt = sns.lineplot(x=x_points, y=y_points)
plt.set(xlabel="# episodes", ylabel="reward")
[Text(0, 0.5, 'reward'), Text(0.5, 0, '# episodes')]
         -240
         -260
         -280
      -320
-320
         -340
         -360
         -380
                       1000
                                2000
                                         3000
                                                   4000
                                                            5000
                                   # episodes
```

SARSA(ALPHA=0.5 GAMMA=0.5 EP=5100)

▼ And the rewards

```
[23] x_points = np.arange(LOG_INTERVAL, NR_EPISODES + 1, LOG_INTERVAL)
y_points = REMARDS

plt = sns.lineplot(x=x_points, y=y_points)
plt.set(xlabel="# episodes", ylabel="reward")

[Text(0, 0.5, 'reward'), Text(0.5, 0, '# episodes')]

-50
-100
-150
-200
-350
-400
-450
0 1000 2000 3000 4000 5000
# episodes
```

▼ 4.3 Results (35p)

Q-Learning(ALPHA = 0.4 GAMMA= 0.7 EP=7000)

And the rewards

x_points = np.arange(LOG_INTERVAL, NR_EPISODES + 1, LOG_INTERVAL)
y_points = REWARDS

plt = sns.lineplot(x=x_points, y=y_points)
plt.set(xlabel="# episodes", ylabel="reward")

[. [Text(0, 0.5, 'reward'), Text(0.5, 0, '# episodes')]

-200
-350
-300
-300
0 1000 2000 3000 4000 5000 6000 7000
episodes

SARSA(ALPHA = 0.4 GAMMA= 0.7 EP=7000)



Q-Learning(ALPHA = 0.3 GAMMA = 0.8 EP=7000 DECAY_EPS=0.85)

```
T And the rewards

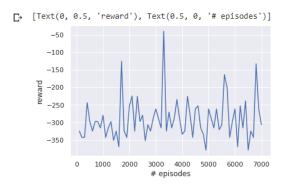
[17] x_points = np.arange(LOG_INTERVAL, NR_EPISODES + 1, LOG_INTERVAL)
y_points = REWARDS

plt = sns.lineplot(x=x_points, y=y_points)
plt.set(xlabel="# episodes", ylabel="reward")

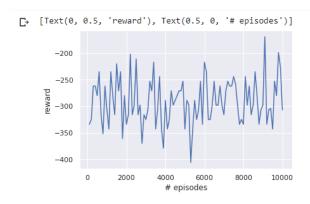
[Text(0, 0.5, 'reward'), Text(0.5, 0, '# episodes')]

-200
p-250
-350
0 1000 2000 3000 4000 5000 6000 7000
# episodes
```

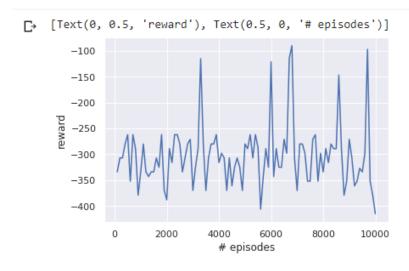
SARSA(ALPHA = 0.3 GAMMA = 0.8 EP=7000 DECAY_EPS=0.85)



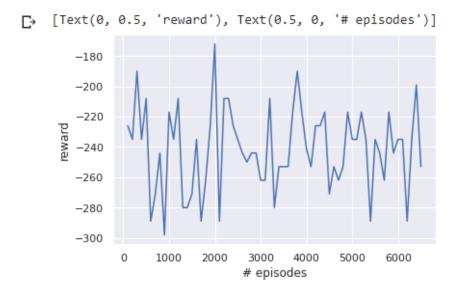
Q-Learning(ALPHA = 0.3 GAMMA = 0.8 EP=10000 DECAY_EPS=0.75 EPSILON=0.5)



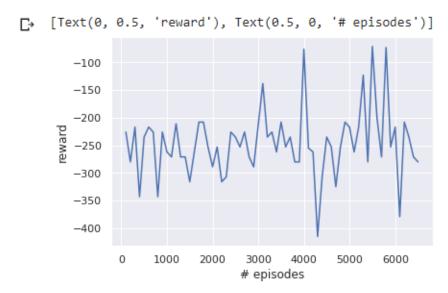
SARSA(ALPHA = 0.3 GAMMA = 0.8 EP=10000 DECAY_EPS=0.75 EPSILON=0.5)



Q-Learning(ALPHA = 0.8 GAMMA=0.4 EP = 6500)



SARSA(ALPHA = 0.8 GAMMA=0.4 EP = 6500)



Analiza:

Cu cat algoritmii sunt rulati pe mai multe episoade,cu atat algoritmii invata mai mult,dar nu inseamna ca mai bine sau eficient in mod neaparat.

Daca learning rate-ul este mare(alpha) atunci ar aparea o problema sezibabila in invatare, si s-ar observa si pe grafice, dar in cazul nostru prea este posibil deoarece avem acea "atenuare" facand (1-alpha).

Implementare:

Implementarea a constat in crearea unei functii choose_action care intoarce o actiune cu o probabilitate eps,altfel se maximizeaza utilitatea in actiunea curenta.

- Q-Learning: primul pas este alegerea unei actiuni, se updateaza starea curenta cu noua valoare dupa formula predate la curs si se trece la new state.
- SARSA: primul pas este alegerea unei actiuni,se updateaza starea curenta cu noua valoare dupa formula predate la curs si se trece la new state. Diferit fata de Q-Learning este faptul ca aici ne folosim de starea urmatoare atunci cand calculam valoare starii curente.