UNIVERSITATEA TEHNICĂ “GHEORGHE ASACHI” DIN IAȘI

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

DOMENIUL CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

SPECIALIZAREA TEHNOLOGIA INFORMAȚIEI

INTELIGENȚĂ ARTIFICIALĂ

Algoritmul Q-Learning pentru un joc de tip Frozen Lake

Coordonator,

Prof. dr. Florin Leon

Studenți,

Bahnaru Alexandru-Georgian, 1409A

Bulgaru Vlad-Andrei, 1409A

An universitar 2024-2025

Cuprins

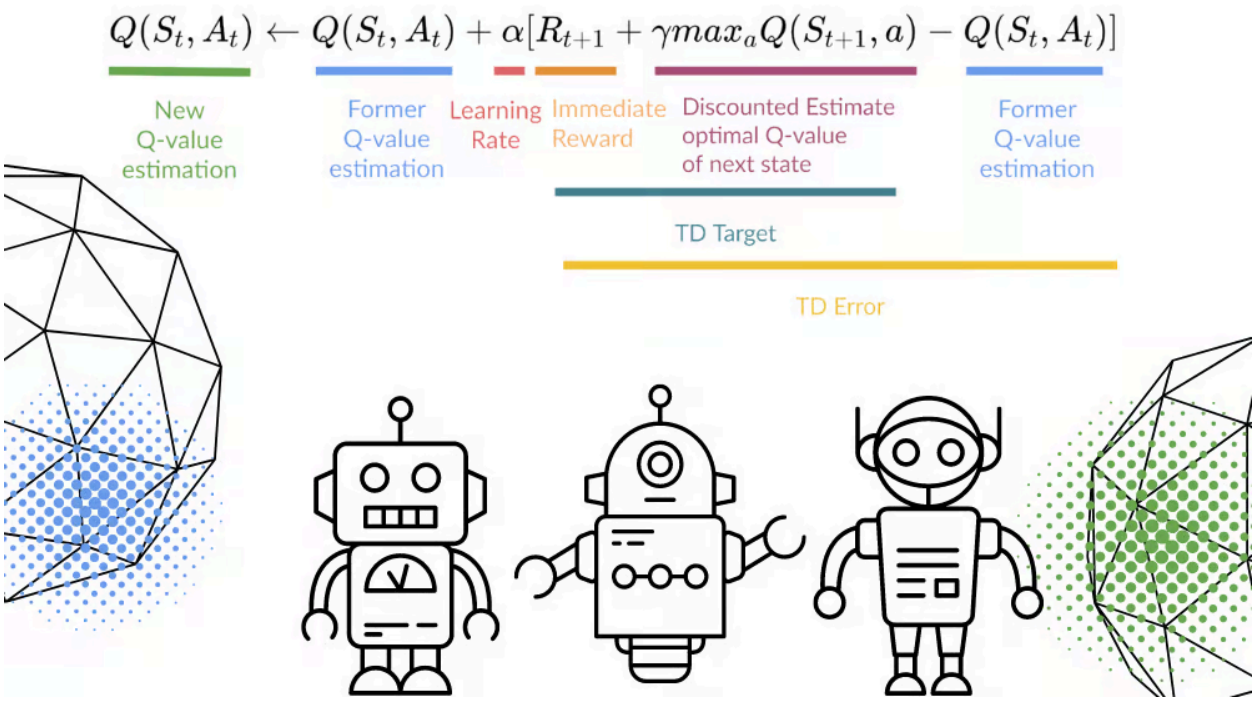
Descriere

Problema Frozen Lake este o problemă din domeniul învățării prin întărire, care se poate vizualiza ca un joc într-un lac înghețat. Un agent trebuie să parcurgă de la punctul de start ('S') până la destinația finală ('G'). Acesta va evita gropile ('H') și va încerca să treaca prin celula de recompensă ('R'). Lacul este împărțit într-o grilă de celule, fiecare celulă având un anumit tip: 'F' pentru apă înghețată, 'H' pentru groapă, 'S' pentru start, 'R' pentru recompensă și 'G' pentru destinație.

Scopul agentului este de a învăța să navigheze în acest mediu folosind algoritmul Q-Learning, un algoritm de învățare prin întărire. Agentul va explora mediul, va învăța din recompensele obținute și va dezvolta o politică optimă de acțiuni pentru a ajunge la destinație, evitând obstacolele.

Algoritmul Q-Learining

Q-learning este un algoritm de învățare prin întărire, fără model, utilizat pentru a determina politica optimă de selectare a acțiunilor pentru orice proces decizional Markov dat. În Q-learning, agentul învață să ia decizii, învățând recompensele așteptate pe termen lung asociate cu fiecare acțiune într-o anumită stare, fără a necesita un model al mediului. "Q" înseamnă calitate, iar calitatea reprezintă cât de valoroasă este acțiunea în maximizarea recompenselor viitoare.
Formula de actualizare a algoritmului este:



Unde:

- Q(s,a) este valoarea acțiunii a în starea s
- α este rata de învățare
- γ este factorul de discount
- Rt+1 este recompensa obținută după efectuarea acțiunii a
- St+1 este noua stare obținută după acțiunea a

Mai jos sunt câteva terminologii utile pentru a înțelege fundamentele Q-learning:

- State(s): poziția actuală a agentului în mediu.
- Action(s): un pas făcut de agent într-o anumită stare.
- Reward(s): pentru fiecare acțiune, agentul primește o recompensă și o penalizare.
- Episode(s): sfârșitul etapei, în care agenții nu pot întreprinde noi acțiuni. Se întâmplă atunci când agentul a atins obiectivul sau a eșuat.
- Q(St+1, a): valoarea Q optimă preconizată a efectuării acțiunii într-o anumită stare.
- Q(St, At): este estimarea curentă a Q(St+1, a).
- Tabelul Q: agentul menține tabelul Q al seturilor de stări și acțiuni.
- Diferențe temporale (TD): utilizate pentru a estima valoarea așteptată a Q(St+1, a) utilizând starea și acțiunea curentă și starea și acțiunea anterioară.

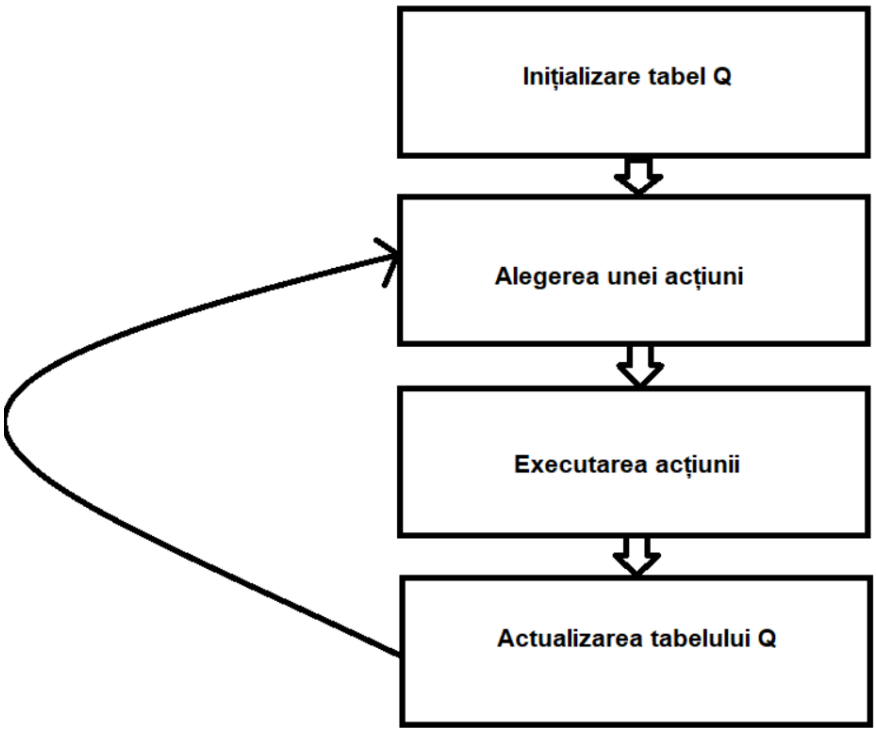
Modalitatea de rezolvare

Agentul trebuie să traverseze un labirint și să ajungă la punctul final. Există găuri în lacul înghețat, iar agentul nu se poate deplasa decât o singură piesă la un moment dat. Dacă agentul

cade în gaură, acesta este mort. Agentul trebuie să ajungă la punctul final în cel mai scurt timp posibil.
Mai jos sunt pașii implicați în Q-learning:

1. **Definirea spațiului de stare și de acțiune:** Spațiul stărilor reprezintă stările posibile în care se poate afla agentul, în timp ce spațiul acțiunilor reprezintă acțiunile posibile pe care le poate întreprinde agentul.
2. **Inițializarea tabelului Q:** Tabelul Q este o matrice care stochează recompensa așteptată pentru fiecare acțiune în fiecare stare. Aceasta este setată inițial la 0.
3. **Observarea stării:** Agentul observă starea actuală a mediului.
4. **Alegerea unei acțiuni:** Agentul selectează o acțiune pe baza unui compromis explorare-exploatare. Echilibrul explorare-exploatare este un echilibru între alegerea acțiunilor care au produs recompense mari în trecut (exploatare) și încercarea de noi acțiuni pentru a afla mai multe despre mediu (explorare).
5. **Executarea acțiunii:** Agentul execută acțiunea aleasă în mediu și observă recompensa primită.
6. **Actualizarea tabelului Q:** Tabelul Q este actualizat folosind recompensa observată, utilizând regula de actualizare Q-learning. Regula de actualizare Q-learning actualizează recompensa așteptată pentru acțiunea aleasă în starea observată pe baza recompensei observate și a recompensei așteptate pentru următoarea pereche stare-acțiune.
7. **Se repetă etapele 3-6 până la convergență:** Etapele 3-6 se repetă până când valorile Q converg către valorile optime, care reprezintă recompensa maximă așteptată pe termen lung pentru fiecare acțiune în fiecare stare.

În final, odată ce tabelul Q a convers la valorile sale optime, agentul îl poate utiliza pentru a selecta cea mai bună acțiune de întreprins în orice stare dată, alegând acțiunea cu cea mai mare recompensă așteptată.



Implementare

Definirea mediului

```python
class FrozenLake:
    def __init__(self, grid_size=7):
        self.grid_size = grid_size
        self.reset()
        self.actions = {
            0: (-1, 0),  # sus
            1: (1, 0),  # jos
            2: (0, -1),  # stanga
            3: (0, 1)  # dreapta
        }
```

Mediul este reprezentat printr-o grilă de dimensiune 7x7, iar fiecare celulă poate fi: “S” pentru Start, “F” pentru apa înghețată, “H” pentru groapă, “R“ pentru recompensă și G” pentru destinație.

Resetarea mediului

```python
def reset(self):
    self.lake = [
        ['S', 'F', 'F', 'F', 'F', 'F', 'F'],
        ['F', 'H', 'F', 'F', 'H', 'F', 'F'],
        ['F', 'F', 'F', 'H', 'F', 'F', 'H'],
        ['F', 'H', 'F', 'F', 'R', 'H', 'F'],
        ['F', 'F', 'F', 'H', 'F', 'F', 'F'],
        ['F', 'H', 'F', 'F', 'H', 'F', 'F'],
        ['F', 'F', 'H', 'F', 'F', 'F', 'G']
    ]
    self.state = (0, 0)
    self.done = False
    return self.get_state_index()
```

Mediul este resetat la fiecare episod. Agentul începe din celula 'S' și se află într-o stare inițială de (0, 0).

Pasul în Mediul Frozen Lake

```python
def step(self, action):
    move = self.actions[action]
    new_row = max(0, min(self.grid_size - 1, self.state[0] + move[0]))
    new_col = max(0, min(self.grid_size - 1, self.state[1] + move[1]))
    self.state = (new_row, new_col)

    cell = self.lake[new_row][new_col]
    if cell == 'H':
        print("Ai cazut intr-o groapa")
        self.done = True
        reward = -1
    elif cell == 'G':
        print("Ai ajuns la destinatie")
        self.done = True
        reward = 1
    elif cell == 'R':
        reward = -0.05
    else:
        reward = -0.1

    return self.get_state_index(), reward, self.done
```

Fiecare acțiune efectuată de agent modifică starea acestuia pe hartă. Dacă agentul ajunge într-o groapă, va pierde, iar dacă ajunge la destinație, va câștiga. Dacă ajunge într-o celulă de apă înghețată, primește o recompensă mică, iar dacă ajunge în celula unde se află morcovul, va primi o recompensă mai mare.

Algoritmul de Antrenare Q-Learning

```python
alpha, gamma, epsilon = 0.8, 0.95, 0.1
num_episodes = 3000
grid_size = 7
num_states = grid_size * grid_size
num_actions = 4
Q = np.zeros((num_states, num_actions))

env = FrozenLake(grid_size=grid_size)
for _ in range(num_episodes):
    state = env.reset()
    done = False
    while not done:
        if random.uniform(a: 0, b: 1) < epsilon:
            action = np.random.choice(num_actions)
        else:
            action = np.argmax(Q[state])
        next_state, reward, done = env.step(action)
        Q[state, action] += alpha * (reward + gamma * np.max(Q[next_state]) - Q[state, action])
        state = next_state
```

În această secțiune, agentul este antrenat folosind algoritmul Q-Learning. Pentru fiecare episod, agentul explorează mediul, ia acțiuni și actualizează valorile Q folosind formula de actualizare. Parametrul alpha reprezintă rata de învățare, gamma reprezintă factorul de discount, iar epsilon reprezintă probabilitatea de a alege o acțiune aleatorie în loc să urmeze politica curentă. Antrenarea se realizează pe un număr de episoade de 3000 și pentru fiecare episod se resetează mediul, agentul fie execută acțiuni random sau bazate pe politica curentă și valoarea Q este actualizată. Procesul va continua până când jocul este gata.

Interfața Grafică (GUI)

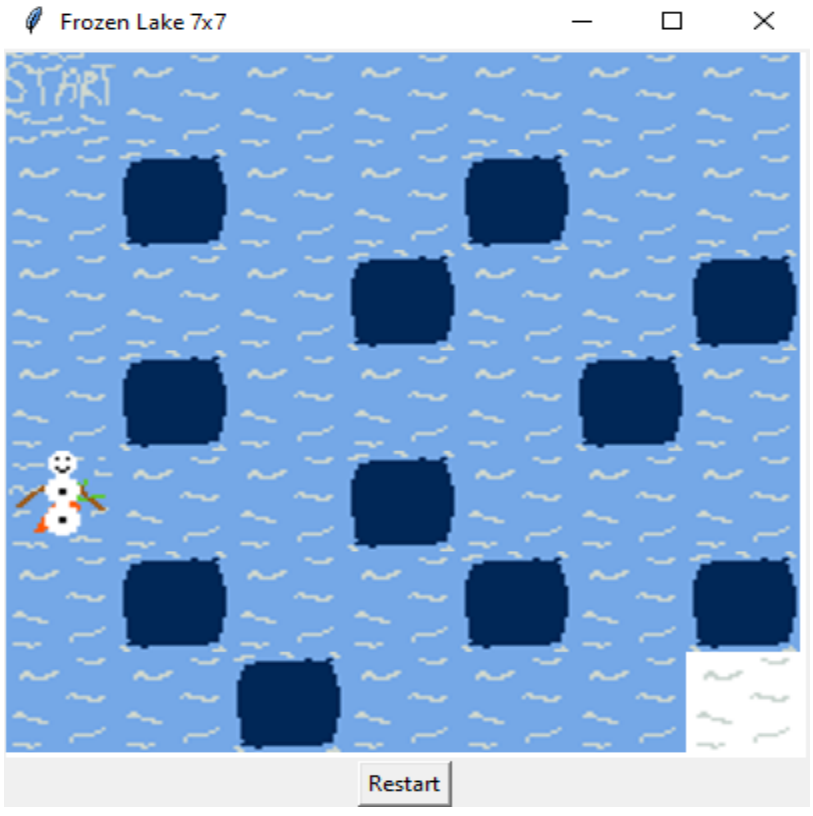
```
class FrozenLakeGUI:
    def __init__(self, root, env, Q, image_paths):
        self.root = root
        self.env = env
        self.Q = Q
        self.canvas_size = 400
        self.cell_size = self.canvas_size // env.grid_size
        self.canvas = tk.Canvas(root, width=self.canvas_size, height=self.canvas_size, bg="white")
        self.canvas.pack()
        self.restart_button = tk.Button(root, text="Restart", command=self.restart)
        self.restart_button.pack()

        self.images = {}
        for key, path in image_paths.items():
            image = Image.open(path)
            resized_image = image.resize((self.cell_size, self.cell_size))
            tk_image = ImageTk.PhotoImage(resized_image)
            self.images[key] = tk_image

        self.restart()
```

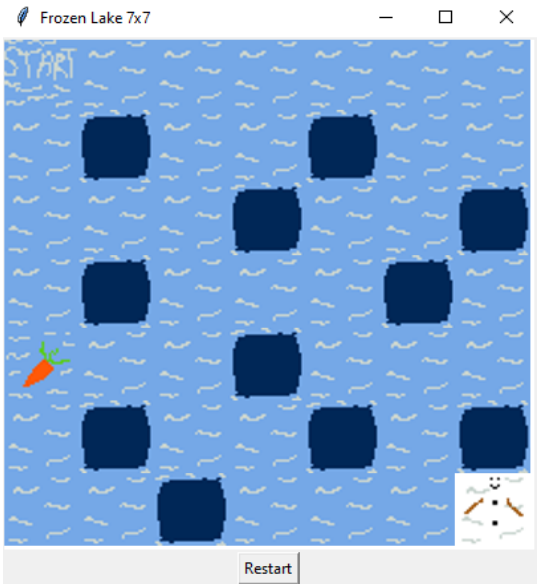
```python
def draw_lake(self):
    self.canvas.delete("all")
    for i in range(self.env.grid_size):
        for j in range(self.env.grid_size):
            x, y = j * self.cell_size, i * self.cell_size
            cell = self.env.lake[i][j]

            if cell == 'S':
                self.canvas.create_image( *args: x, y, anchor=tk.NW, image=self.images['start'])
            elif cell == 'H':
                self.canvas.create_image( *args: x, y, anchor=tk.NW, image=self.images['hole'])
            elif cell == 'R':
                self.canvas.create_image( *args: x, y, anchor=tk.NW, image=self.images['carrot'])
            elif cell == 'F':
                self.canvas.create_image( *args: x, y, anchor=tk.NW, image=self.images['ice'])
            elif cell == 'G':
                self.canvas.create_image( *args: x, y, anchor=tk.NW, image=self.images['goal'])

    x, y = self.env.state[1] * self.cell_size, self.env.state[0] * self.cell_size
    self.canvas.create_image( *args: x, y, anchor=tk.NW, image=self.images['snowman'])
```

```python
def restart(self):
    self.env.reset()
    self.done = False
    self.run_agent()


2 usages
def run_agent(self):
    if not self.done:
        state_index = self.env.get_state_index()
        action = np.argmax(self.Q[state_index])
        _, _, self.done = self.env.step(action)
        self.draw_lake()
        self.root.after(300, self.run_agent)
    else:
        print("Episod terminat")
```

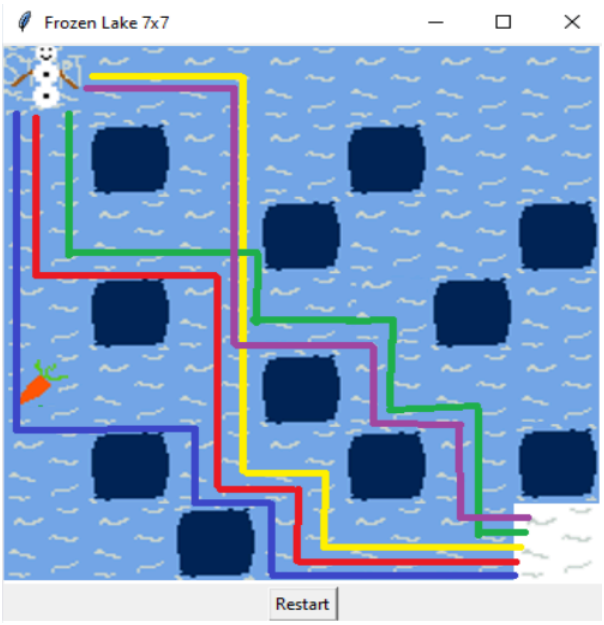
Aceasta este o interfață grafică creată cu ajutorul bibliotecii Tkinter, care vizualizează starea mediului și mișcările agentului. La fiecare pas, mediul este actualizat și agentul este deplasat conform politicii învățate. Elementele cheie ale acestei interfețe sunt canvas-ul, butonul de restart și imaginile din fiecare celulă din matrice.

Rezultatele obținute prin rularea programului





După cum putem observa, există 5 drumuri de aceeași lungime minimă (12), dar agentul va trece doar pe drumul în care se află morcovul.



Concluzii

Așadar, algoritmul Q-Learning a fost implementat cu succes pentru a rezolva problema Frozen Lake. Agentul învață treptat să navigheze prin lac, evitând gropile și ajungând la destinație. Performanța algoritmului se îmbunătățește pe măsură ce numărul de episoade crește, iar agentul devine din ce în ce mai bun la alegerea acțiunilor corecte.

Acest proiect demonstrează eficiența algoritmului Q-Learning în medii de învățare prin întărire, dar și importanța parametrilor precum rata de învățare și factorul de discount.

Împărțirea sarcinilor de lucru în echipă

Bahnaru Alexandru-Georgian
Bulgaru Vlad-Andrei

Bibliografie

https://utsavdesai26.medium.com/mastering-q-learning-hands-on-examples-and-key-concepts-5e610d91a12b
https://www.datacamp.com/tutorial/introduction-q-learning-beginner-tutorial?utm_source=google&utm_medium=paid_search&utm_campaignid=19589720824&utm_adgroupid=157156376071&utm_device=c&utm_keyword=&utm_matchtype=&utm_network=g&utm_adpostion=&utm_creative=720362650444&utm_targetid=dsa-2218886984380&utm_loc_interest_ms=&utm_loc_physical_ms=1011828&utm_content=&utm_campaign=230119_1-sea~dsa~tofu_2-b2c_3-row-p2_4-prc_5-na_6-na_7-le_8-pdsh-go_9-nb-e_10-na_11-na-bfcm24&gad_source=1
https://www.freecodecamp.org/news/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc/
https://edu.tuiasi.ro/pluginfile.php/49458/mod_resource/content/9/IA13_Intarire.pdf
https://towardsdatascience.com/reinforcement-learning-101-q-learning-27add4c8536d
https://www.piskelapp.com/