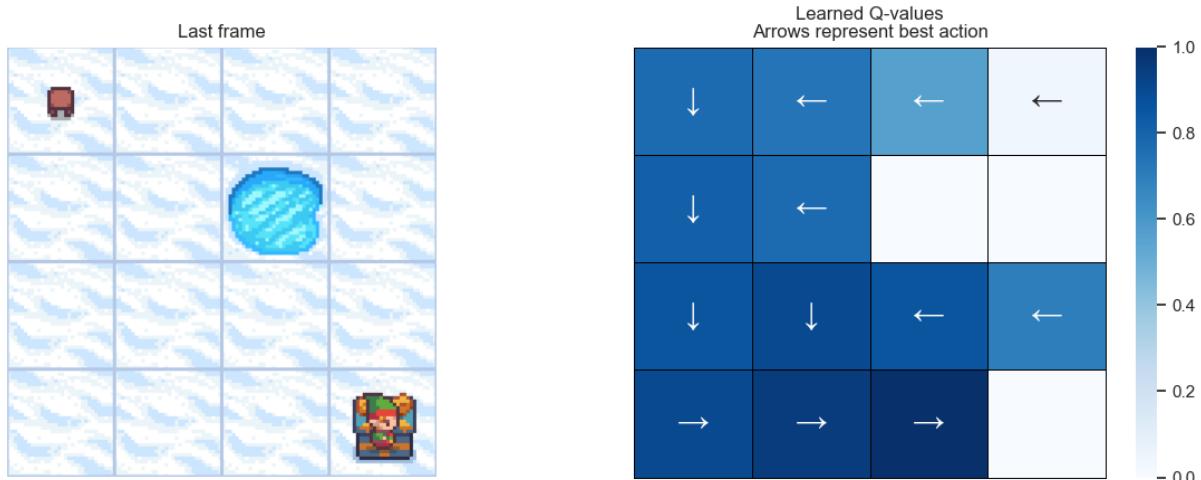


מבוא לבינה מלאכותית

סמסטר חורף תשפ"ה

מטלה 4

תאריך הגשה: 23:55 14.1.2025



הנחיות

- שאלות בנושא מטלה זו יש לשאול דרך המודל, בפורום "מטלה 5".
- הוראות להגשת המטלה מופיעים בסוף מסמך זה.
- הקבצים הנדרשים להרצת הקוד הימן:
 - q_learning.py
 - common.py
 - value_iteration.py
 - plotting.py
 - frozen_lake_experiment.py
 - gamblers_experiment.py
 - requirements.txt
- העבודה להגשה בזוגות בלבד אלא אם כן המגנים קיבלו אישור להגשת שאינה בזוגות.
- הקוד הורץ ונבדק על **WINDOWS** אך יכול לרוץ בסביבת mac או axchon או אבל יתכן וידרשו התאמות.
- פתרון המטלה שתגישיו יבדק מול שאר ההצעות על ידי תוכנת העתקות.
- מי שימצא כי העתיק יכשל בקורס וכן יעבור לועדת משמעת אוניברסיטאית.
- הפרויקט נכתב ויבדק בשפת התכנות python, גרסה 3.10.
- **יש להתקין את החבילות על פי הקובץ requirements.txt.**
 - על מנת לוודא שהסביבה מותקנת במלואה יש להריץ:
pip install "gymnasium[toy-text]"

- מסמך זה בניי באופן הבא: **תיאור המטלה**, **תיאור הבעיה**, **שאלות המטלה** ו**הסבר על המימוש הקיימים** והסביר על **הגשת המטלה**.

תיאור המטלה:

בתרגיל זה אתם מתחקים למש את האלגוריתמים **Value Iteration** ו- **Q Learning** על מנת לפתור שטי בעיות.

בחלק הראשון של התרגיל עליכם למש את האלגוריתם של **Q Learning** כדי לפתור את בעיית ה-**frozen lake** ולאחר מכן עליכם למש את האלגוריתם **Value Iteration** על מנת שתוכלו לפתור את בעיית המהמר (עם שינוי).

בעיית ה-**Frozen Lake**:

בעיה זו הממצאים והפעולות מוגבלים על ידי מספרים מסוימים אט הממצב (על פי המימדים של המפה) וכיוון התנועה של הסוכן (0 עד 3über הכוונים בהם הסוכן מסוגל לנوع).

עליכם למש את הפונקציות הבאות במחלקה **:QLearning**

```
def update(self, state: int, action: int, reward: float,
new_state: int):
```

הfonkzia מקבלת את המצב הנוכחי, ה-**reward** והמצב הבא ומעדכנת את ערך ה-**Q table** בהתאם על פיו משוואת בלמן שנלמדה בכיתה.

```
def select_epsilon_greedy_action(self, state: int) -> int:
```

הfonkzia מקבלת את המצב הנוכחי ומוחזירה את הפעולה לביצוע על פי אלגוריתם **epsilon greedy**.
שימוש לב, אם ישן מספר פעולות אופטימליות מומלץ לבצע breaking tie רנדומלי.

```
def train_episode(self, env: gym.Env) -> Tuple[float, int]:
```

הfonkzia מקבלת **instance** של הסביבה ומאננת את הסוכן על ה-**episode**. הfonkzia מוחזירה את ה-**reward** שהתקבל ב-**episode** **הfonkzia צריכה לחוץ עד אשר מגיעי למצב טרמינלי**. שימוש לב, פעולה זו *** לא*** מקבלת מספר צעדים וריצה עד אשר הסוכן נפלס או עד אשר המשימה הצלחה והתקבל **reward**.

שימוש לב, הfonkzia (**env.step(action)**) מבצעת צעד יחיד על הסביבה. ערכי החזרה של פונקציה זו הם:

```
new_state, reward, terminated, truncated, info =  
env.step(action)
```

כאשר המשתנה **truncated** מצין האם האינטראקציה עם הסביבה הסתיימה בצורה פתאומית (כישלון המשימה או כישלון בסביבה וכו') והערך **terminated** אומר האם המשחק הסתיים (בהצלחה). כל מנת לדעת האם ה-**episode** הסתיים, בדקו האם מתקיים כי **terminated** או **truncated** מקבלים ערך **True**.

```
def run_environment(self, env: gym.Env, num_episodes: int) ->
    Tuple[List[float], List[int]]:
    pass
```

הfonkziah מריםה על הסביה את מספר ה-*episodes* שנותן בפרמטר.

על הפונקציה להחזיר את ה-*rewards* עבור כל ה-*episode* כמו גם את מספר הצעדים שבוצעו בכל *episode*.

בעית ה-Gambler

בשאלה זו עלייכם למש את אלגוריתם *value iteration* על גורסה חדשה של בעית המהמר.



בעית המהמר מוגדרת באופן הבא:

מהמר מקבל חזדנות להמר על התוצאות של רצף הטלות מטבע. אם המטבע נוחת על צד "עץ", הוא זוכה במספר דולרים כמספר הסכום שהימר; אם נוחת על צד "פל", הוא מפסיד את הסכום שהימר. המשחק מסתיים כאשר המהמר מנצח ומגיע למטרה של 100 דולר, או מפסיד כאשר נגמר לו הבספ.

בכל הטלה, המהמר חייב להחליט איזה חלק מההון שלו להמר, במספרים שלמים של דולרים. בעיה זו יכולה להיות מנוסחת כ-MDP-סופי, אפיוזדי ולא מוזל.

המצב הוא כמות הבספ הנוכחית של המהמר. $s \in \{1, 2, \dots, 99\}$,

הפעולות הן סכומי ההימור. $a \in \{0, 1, \dots, \min(s, 100) - s\}$, התגמול הוא אפס בכל המעברים פרט לאלה שבהם המהמר מגיע למטרה שלו, אז הוא +1.

עבור הבעיה שלנו אנחנו נניח כי במקומות מטבע עם שני צדדים, למהמר יש קוביה שלוש פאות עם הערכים 1,3,6 כאשר התוצאות של הטללה הן כדלהלן:

1. כאשר יצא 1 – המהמר מפסיד את ההימור שהוא שם.
2. כאשר יוצא 3 – המהמר לא מפסיד ולא מרוויח.
3. כאשר יוצא 6 – המהמר מרוויח את הסכום שהימר עליו.

عليיכם למש את הפעולות הבאות במחלקה *ValueIteration*:

```
def calculate_q_values(self, current_capital: int,
    value_function: np.ndarray, rewards: np.ndarray) ->
    np.ndarray:
```

הfonkizia מחשבת את ה- `value` בהינתן המצב הנוכחי, **אוף-h-rewards** (מוגדרים להיות אפס לכל אחד מהמצבים פרט למצב האחרון שהוא reward של 1).

שימוש לב – אתם מקבלים כחלק מהקלט לחלוקת את ההסתברויות של ליפול על 3, ליפול על 1 וליפול על 6. עלייכם להשתמש בהסתברויות הללו על מנת לחשב את הערכי-[a, s] Q כאשר ה- s הינו המצב הנוכחי (`current_capital`) והיתן בקלט לפונקציה.

עליכם למשג גם את הפונקציה

```
def value_iteration_for_gamblers(self) -> Tuple[np.ndarray, np.ndarray]:
```

אשר מרים את תחילת ה- `value iteration` עד להתקנות.

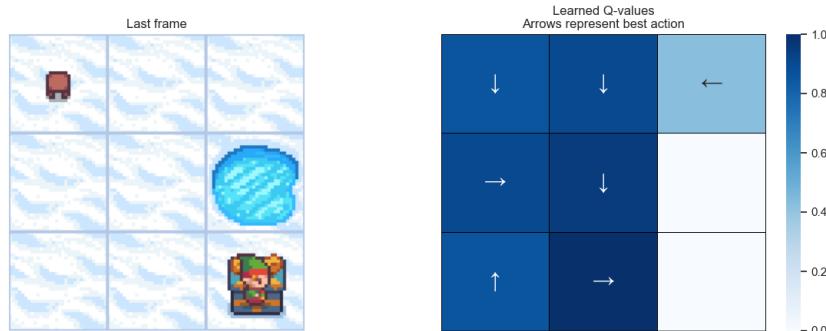
שימוש לב, על הפעולה להחזיר את ה- `values` כמו גם את ה- `policy` האופטימלי שחוושב. מצופה להחזיר באופן הבא:

`return policy, V`

שימוש לב, ישנים 101 ערכים אפשריים ל- `rewards` – 0 עד 100 * כולל* .

בדיקות נכונות:

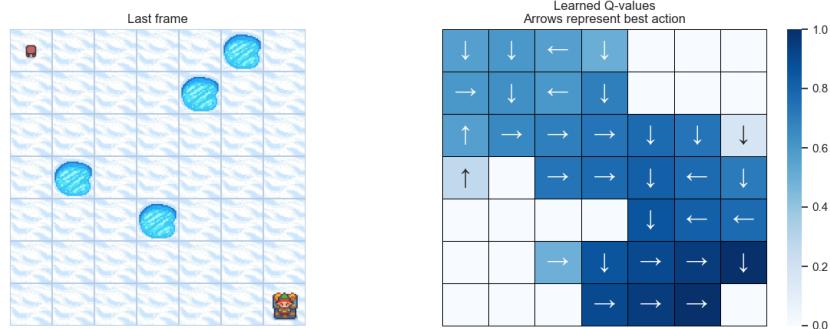
בקבצים `frozen_lake_experiment.py`-ו `gamblers_experiment.py` מצוים תרחישים אשר ידקו את נכונות האלגוריתם שאתם בותחים.



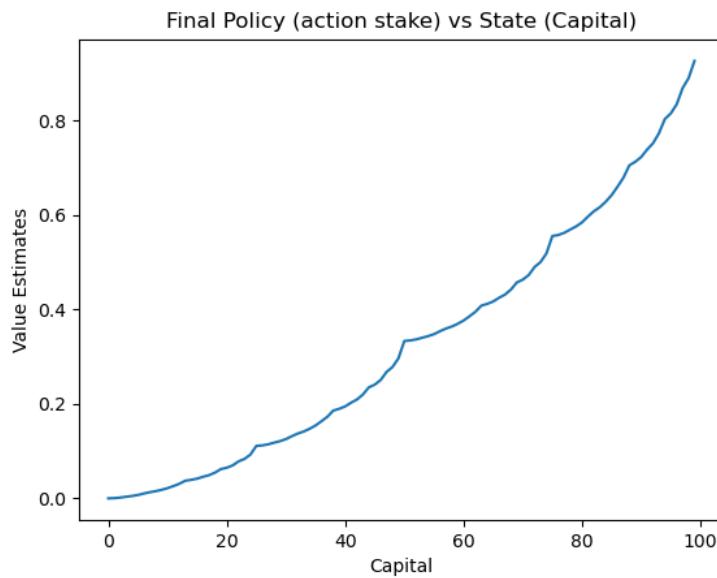
עבור הרצת הסוכן על בעיית ה- `frozen lake` מצופה מכם לקבל תוצאה כזו לאחר סיום הריצה.

עבור בעיית ה- `frozen lake` מומלץ לנסות את הפתרון שלכם על מספר גדול מפה שונים (התרגיל מאוחול עם גודל מפה 3 – משתנה `size_map`) וליזדאו כי בכל הנסיבות אתם מקבלים פתרון תקין וכי הסוכן שלכם מגיע למטרה שלו וכי ה- `policy` שמתקיים ביצירור נראה תקין.

ניסוי עבר גודל מפה 6
map_size=6



עבור בעיית המהמר עם הפרמטרים שבקובץ מצופה מכן לקבל גרפ' כדלהלן:



הגשת המטרה:

המטרה תיבדק ב-`lql`, הקבצים עליהם להגיש הם `learning.py` ו-`value_iteration.py`. הקוד שלכם יעבור בדיקות אוטומטיות על תרחישים בדיקה שמדוים את התறחישים שאתם מראים פה.

הסבירה מראיה את הקוד שלכם על מספר תרחישים וכן הרצאה אמרה לחת מספר שניות.

בצלחה 😊