מבוא למערכות תבוניות, חכמות וקוגניטיביות

פרויקט גמר

ישראלה מגירא
209015817

מבוא

בפרויקט זה המטרה היא **לבנות סוכן כללי שיודע להתמודד עם סוגים שונים של עולמות ובעיות.** האתגר העיקרי נבע מזה שזהו סוכן כללי – כלומר, בתהליך כתיבת הקוד **לא ניתן להסתמך על שום מידע**.

בחרתי לפצל את העולמות ל-2 סוגים: עולם **דטרמיניסטי** ועולם **לא-דטרמיניסטי**.

- **<u>עולם דטרמיניסטי</u>** – בעולם דטרמיניסטי הכול ידוע מראש. כל פעולה שהסוכן יבחר לעשות, תקרה בוודאות. לכן, בעולם כזה מתאים לבצע **תכנון מראש (planning)**. השתמשתי ב - **Plan Dispatcher** של **PDDL**.
- **<u>עולם לא-דטרמיניסטי</u>** – בעולם לא דטרמיניסטי לא הכול ידוע מראש. כאשר הסוכן בוחר פעולה **יש הסתברות p** שהפעולה שבחר אכן תתבצע, **והסתברות 1-p** שתתבצע פעולה אחרת. בנוסף, ישנם מצבים נסתרים שמתגלים רק כאשר מגיעים למצב מוגדר כלשהו. לכן, בעולם כזה מתאים לבצע **למידה**. למידה יכולה לעזור ב-2 היבטים:
    - למידה **מוכוונת מטרה** עבור בעיה ספציפית.
    - **למידת העולם** והכרת הסביבה של הסוכן.

שיטות ואלגוריתמים

עבור למידה השתמשתי ב- **Reinforcement Learning.** בחרתי בשיטה זו, כי לדעתי עבור סוכן כללי **שצריך להתמודד עם כל עולם ובעיה אפשריים**, זו השיטה המתאימה ביותר. היתרון של למידת חיזוקים הוא שבשיטה זו ניתן לשלוט בכל שלב האם אנו **חוקרים את העולם או מנצלים את הידע שכבר רכשנו**. השילוב הזה, של **explore & exploit** מאפשר לנו להתמודד עם עולמות ובעיות שהסוכן לא ראה מעולם, ובנוסף לשמור את הידע שנלמד בכל שלב.

Q-Learning:

השתמשתי ב- **Q-Leaning** על מנת לבצע למידה מוכוונת מטרה. כלומר, בהינתן עולם ובעיה ספציפיים, בעזרת חיזוקים מתאימים הסוכן ילמד את הדרך הטובה ביותר להשגת המטרה.

Algorithm 1 Q-LEARNING

1: Initialize $s=s_0$
2: **Loop** until reached all goals
3: Choose an action
4: Observe r,s'
5: $Q(s,a) = Q(s,a) + \alpha[r + \gamma * \max Q(s',a')]$
6: **End loop**
7. Backward update of Q table

<u>בחירת פעולה:</u>

באמצעות הגרלת מספר נקבע האם לעשות פעולה רנדומלית או להתבסס על הטבלה. על מנת שבשלבים הראשונים הסוכן יוכל לחקור את העולם, ועם הזמן להיעזר יותר ויותר בידע שהוא רכש, נעדכן בכל הרצה את ערכי ה- **learning rate** וה- **exploration rate** בהתאם.

<u>עדכון ה- Q-table:</u>

עדכון הטבלה רק בזמן הלולאה, כפי שיש בעדכון רגיל, לא היה מספיק משמעותי ולכן הוספתי **עדכון לאחור**. כלומר, במהלך הלמידה הסוכן שומר את כל הצעדים שעשה,

וכשמגיעים למטרה הסופית חוזרים על כל הצעדים הללו, מהסוף להתחלה, ומעדכנים לפני הנוסחה הבאה:

$$Q(s,a) = r + \gamma * \max Q(s',a')$$

כך, **כל מצב מושפע יותר מהמצבים העתידיים.**

Reward:

חיזוק חיובי	חיזוק שלילי
הגעה למטרה +100	ביקור במצב שכבר היה בו -5
	ביקור במצב הקודם -10
	כל צעד -1

הרצה:

לאחר מספר הרצות למידה, ההרצה הסופית מתבססת על ה-Q-table שנוצר בלמידה ועל גרף מצבים (הגרף מכיל מצב התחלתי, פעולה, והמצב אחרי הפעולה). **ההרצה הסופית מתבצעת באופן הבא, עבור כל מצב:**

Rmax:

לאחר מימוש ה-Q-learning, עלה הצורך באלגוריתם שאינו מוכוון מטרה. כלומר, **אלגוריתם שילמד את הסביבה בלי להתמקד בבעיה או מטרה ספציפית**, בשביל שהסוכן יוכל ללמוד עולם מסוים ולהצליח להריץ באותו עולם בעיות שונות. בנוסף, אלגוריתם זה יכול לסייע בהתמודדות עם עולמות מורכבים, בהם כמות המצבים והפעולות גדולה, **שכן הוא מגביל את מספר הביקורים בכל מצב.**

העקרון של Rmax התאים לצורך הזה. בפרויקט זה, לא הצלחתי לממש את הRmax בצורה מיטבית (הוספתי את הקוד, ללא הרצה) אך בשורות הבאות **אציג את הרעיונות המרכזיים**.

ראשית, נאתחל את טבלת ה-**reward** כך שלכל הערכים יהיה את הערך המקסימלי. נגדיר כמה פעמים אנחנו רוצים לבקר בכל מצב - נקרא למשתנה **m**. נרוץ בלולאה ונבדוק האם ביקרנו באותו מצב כבר m פעמים, אם כן נקרא למצב זה **"מצב מוכר"** ונעדכן את ה **reward** וה- **transition**. אחרת, אנחנו במצב לא מוכר ולכן ה-Reward יהיה מקסימלי. כל פעם, נעדכן את ה **Policy**.

Algorithm 2 R-MAX

```
Input: S ,A, m, Rmax
Initialize s_r = Rmax, all count=0, s is starting state
Loop
    Choose a = π(s)
    Take action a, observe r, s'
    Increment C(s,a,s'),C(s,a)
    Rsum(s,a) += r
    if C(s,a) ≥ m then
        R(s,a) = Rsum(s,a)/ C(s,a)
        for all s' ∈ C(s,a,*) do
            T(s,a,s') = C(s,a,s')/C(s,a)
    else
        R(s,a) = Rmax
        T(s,a,s_r) = 1
    call value-iteration
    s = s'
```

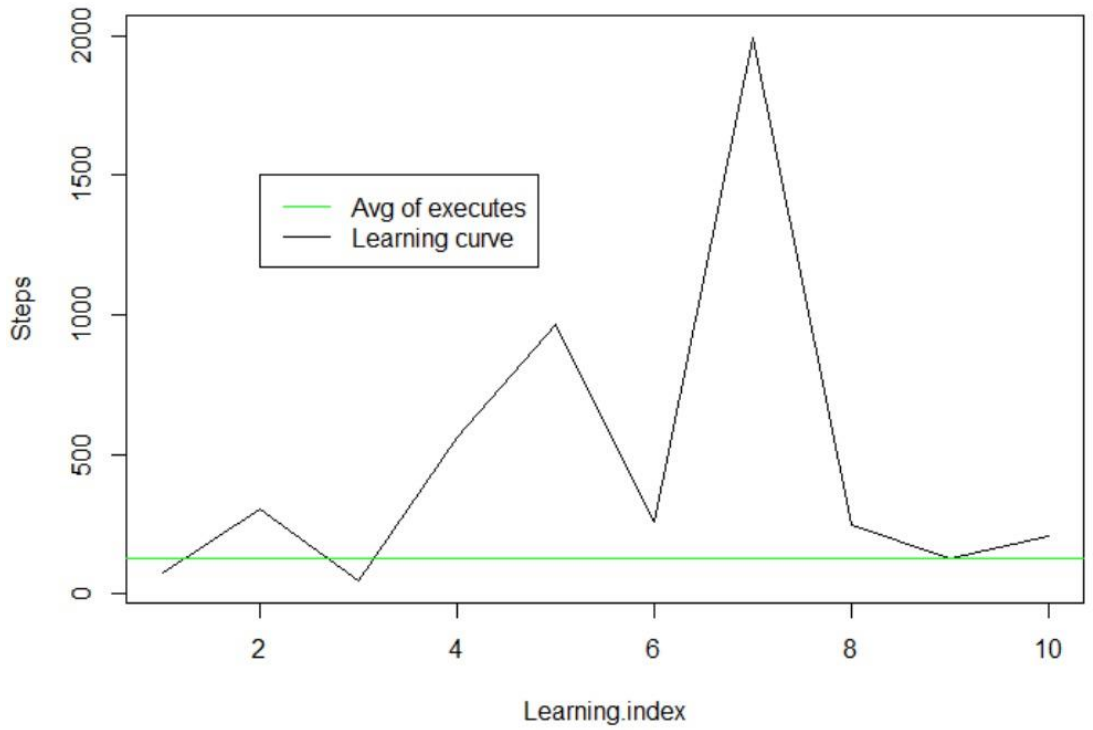
תוצאות ומסקנות

בפרויקט זה **המטרה הייתה לבנות סוכן כללי שיודע להתמודד עם סוגים שונים של עולמות ובעיות.** השיטה בה השתמשתי הינה **Q-learning**, שהיוותה שיטה טובה עבור רוב העולמות שבדקתי, אך לא הספיקה להתמודדות עם עולמות מורכבים כגון הלוויין. במהלך הפרויקט, ניסיתי לממש גם את שיטת **Rmax,** אך מכיוון שההרצה לא הייתה מיטבית, כללתי רק את הרעיונות המרכזיים של השיטה, ללא הרצה.

להלן דוגמאות לתהליך הלמידה שביצע הסוכן בשיטת הQ-learning בעולם המבוך הלא-דטרמיניסטי:

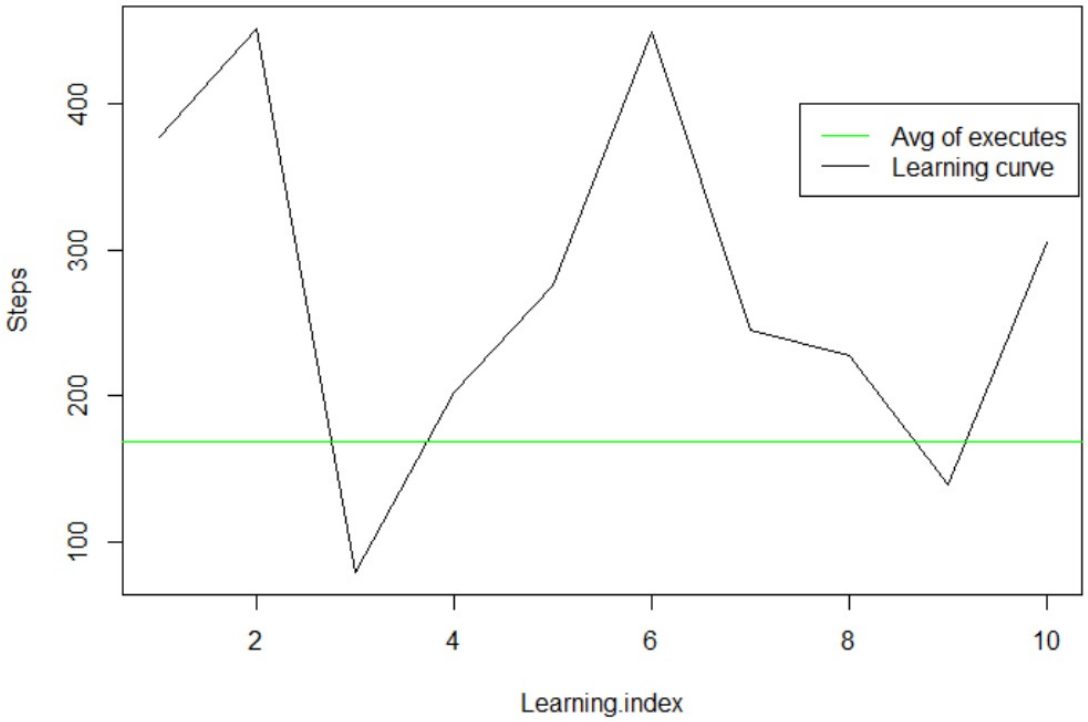
גרף 1: ניסיונות הלמידה של הסוכן בעולם המבוך הלא דטרמיניסטי, עם מטרה אחת

הקו הירוק מסמן את ממוצע שלושת ניסיונות ההרצה - 124



גרף 2: ניסיונות הלמידה של הסוכן בעולם המבוך הלא דטרמיניסטי, עם מטרות מרובות

הקו הירוק מסמן את ממוצע שלושת ניסיונות ההרצה - 124



בגרפים שלעיל, ניתן לראות כי מספר הצעדים בכל ניסיון למידה משתנה, ולעתים אף עולה, בשל הרנדומליות הקיימת בתהליך הלמידה.

ניתן לשפר את ההתמודדות עם עולמות הבעיה באמצעות **מספר פתרונות נוספים,** שחקרתי לגביהם אך לא הספקתי למשם. ראשית**, הרצת הRmax באופן מיטבי** כדי להתמודד עם עולמות מורכבים. שנית, כדי לייצר policy טוב, ניתן למשם **Value Iteration** או **Policy Iteration**.

References

1. https://papers.nips.cc/paper/8484-sample-efficient-deep-reinforcement-learning-via-episodic-backward-update.pdf
2. https://ie.technion.ac.il/~moshet/brafman02a.pdf
3. TEXPLORE: Temporal Difference Reinforcement Learning for Robots and Time, pg.16