בס"ד

תכנון מסלול רובוטי במרחב דו ממדי באמצעות בינה מלאכותית

פרויקט סוף סמסטר בקורס בינה מלאכותית

שגב יצחק איזק
207938085

המכללה האקדמית להנדסה ע"ש סמי שמעון

בס"ד

הבעיה שנבחרה וניתוח הגישות לפתרונה:

הבעיה שבחרתי לעסוק בה היא **תכנון מסלול רובוטי** ,בעיה הקשורה בתחום **למידת חיזוק** ו**תכנון משימות** .בהקשר זה, בחרתי שתי גישות עיקריות לפתרון הבעיה **Q-learning** :ו **Neural-** **Networks**, .

לניתוח הבעיה נתמקד בכמה נושאים שקשורים לתחומים השונים של AI שלמדנו בקורס:

1. **Problem-Solving Agents** - הבעיה של תכנון מסלול רובוטי היא בעיית חיפוש שבה הסוכן (הרובוט) מחפש דרך אופטימלית ממקום למקום. בעיה זו ניתנת לפתרון על ידי סוכנים המבצעים חיפוש במצבים שונים, ומבצעים חיפוש במרחב פתרונות באמצעות אלגוריתמים שונים. במובן זה, הבעיה משתלבת בצורה טובה עם גישות של **חיפוש ללא ידע מוקדם** שמומשו למשל באלגוריתמים כמו BFS וDFS.

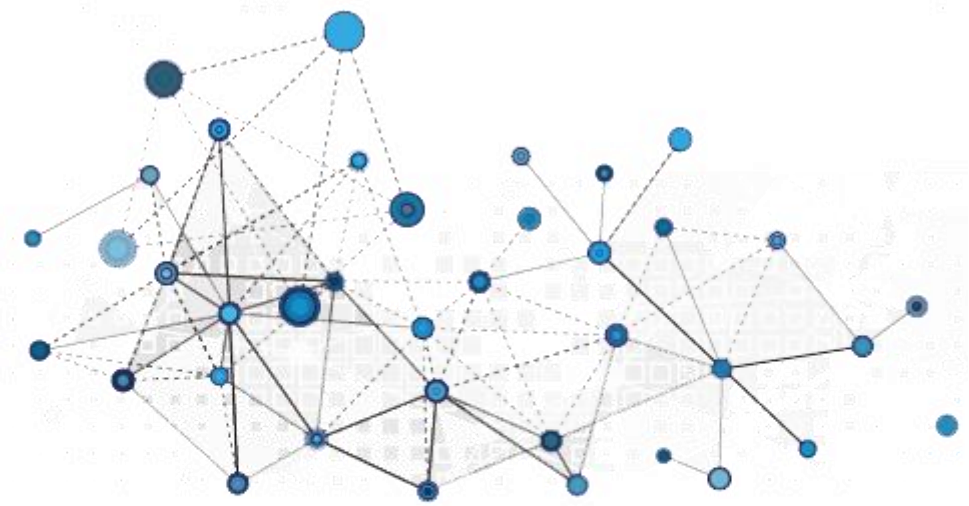
2. **Informed Search and Local Search** - לאור העובדה שלרוב הבעיה כרוכה במרחב חיפוש רחב מאוד, יש צורך בפתרונות של חיפוש מושכל. אלגוריתמים כמו **A*** יכולים לשפר את החיפוש על ידי הערכה של עלות ההגעה למטרה. זה רלוונטי במיוחד כשמדובר בתכנון מסלול רובוטי במצבים דינמיים ומורכבים.

3. **Machine Learning and Q-learning** - בתחום של **למידת מכונה** ,המודלים של-Q learning ו DQN-מציעים פתרון אידיאלי Q-learning .מאפשר לרובוט ללמוד כיצד לבחור פעולות על פי תגמולים עבור בעיה שבה אין לנו גישה ישירה למידע מלא על הסביבה-Q , learning הוא פתרון יעיל שמאפשר לרובוט ללמוד מהתנסות ולהתאים את הפעולות שלו.

4. **Artificial Neural Networks -** רשתות נוירוניות, ובמיוחד ,Deep Q Networks (DQN) מספקות פתרון נוסף שיכול להתמודד עם בעיות יותר מורכבות שבהן יש מספר גדול של מצבים ופעולות. הרשתות הנוירוניות מאפשרות למידה של פונקציות Q על ידי ניתוח של סביבות מורכבות וייצוגם באמצעות שכבות נוירונים.

5. **Planning -** עבור בעיות של תכנון, הרובוטים חייבים לקבוע סדר פעולות ולבחור מה לעשות בהמשך. תכנון פעולות הוא חיוני במיוחד כאשר מדובר בבעיה הדורשת סדר פעולה לא לינארי, כלומר לקבוע אילו צעדים יש לנקוט כדי להגיע למטרה בהצלחה.

בהתאם לנושאים אלו, נבחרו האלגוריתמים **Q-learning**ו **Neural Networks**-לפתרון הבעיה, כל אחד מהם מספק יתרונות שונים בתהליך הלמידה. ה Q-learning-מאפשר למידה ישירה ותגובה מהירה, בעוד ש Deep Q Networks-מביאים יתרון בשימוש בסביבות מורכבות יותר, שבהן יש צורך בהבנת קשרים מסובכים בין המצב והפעולות האפשריות.



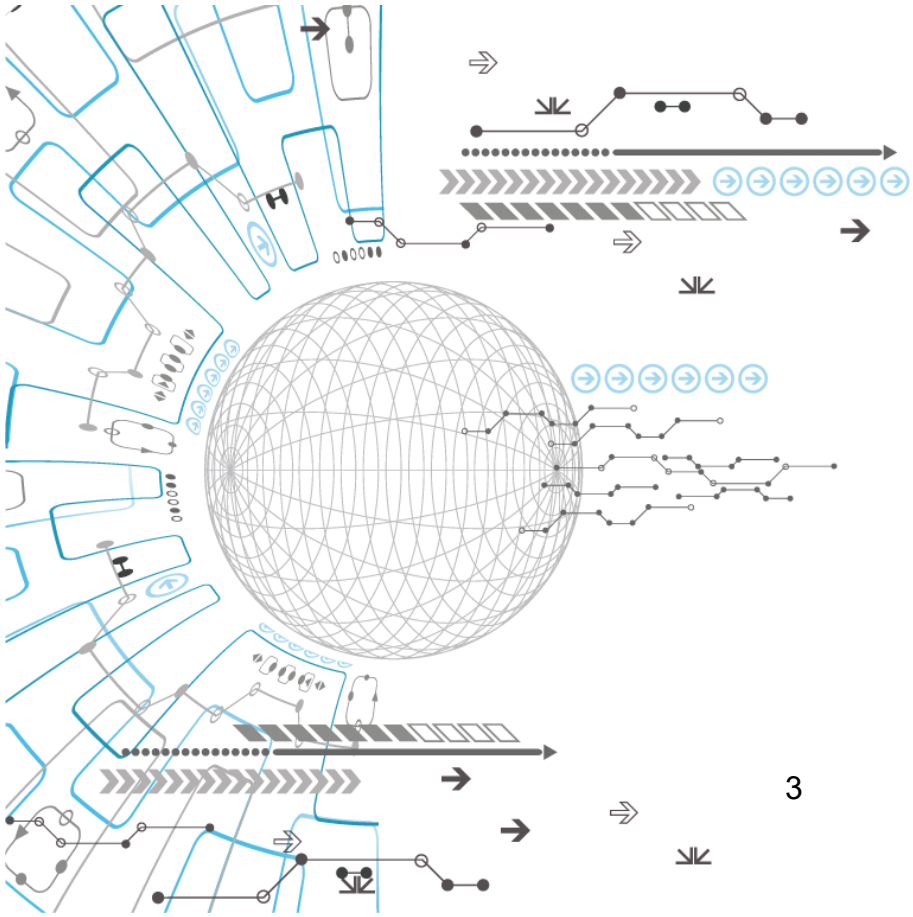
תחומים המתאימים לפתרון הבעיה:

הבעיה של תכנון מסלול רובוטי נחשבת לבעיה **של חיפוש במרחב מצבים** ,כאשר הרובוט חייב למצוא את הדרך היעילה ביותר ממקום אחד למקום אחר. תחום זה מצריך שימוש במודלים המתמודדים עם חיפוש ובחירת פעולות בצורה חכמה. התחומים המרכזיים המופיעים בהקשר של תכנון מסלול רובוטי כוללים:

1. **למידת חיזוק** :**(Reinforcement Learning)** התחום המרכזי לפתרון בעיות כמו תכנון מסלול רובוטי. בלמידת חיזוק, הסוכן (הרובוט) לומד על פי תגמולים ועונשים, מה שיכול להיות מאוד אפקטיבי כאשר אין לנו ידע מלא מראש על הסביבה בה הרובוט פועל.
אלגוריתמים כמו **Q-learning** ו**Deep Q Networks (DQN)**-מאוד פופולריים בהקשר זה.

2. **תכנון** :**(Planning)** תחום נוסף חשוב בתכנון מסלול רובוטי, שבו הרובוט לא רק מגיב למצבים, אלא מתכנן סדר פעולות מראש. תכנון בדרך כלל כולל חיפוש במרחב פתרונות או אלגוריתמים כמו **A*** לחיפוש מושכל.

3. **למידת מכונה** :**(Machine Learning)** שימוש במודלים של רשתות נוירוניות הוא תחום מרכזי במיוחד כאשר מדובר בבעיות של חיזוי התנהגות של סביבות דינמיות ומורכבות.
שימוש ב DQN-מאפשר לרובוט ללמוד ולנבא את הערכים (Q-values) של מצבים ופעולות שונים.



אלגוריתמים/פתרונות מודרניים(State of the Art)

1. **Q-learning**:

- **סקירה ספרותית** Q-learning: הוא אלגוריתם למידת חיזוק שבו הסוכן לומד על ידי תגמולים/עונשים, ללא צורך במידע מקדים על הסביבה. האלגוריתם מבצע עדכון של **טבלת Q** עבור כל מצב ופעולה, כך שהסוכן יבחר את הפעולה שמביאה את התגמול הגבוה ביותר בטווח הארוך.
- **מאמרים רלוונטיים** :מאמרים על Q-learning מדברים על היישום של האלגוריתם בבעיות כמו תכנון מסלול רובוטי, תכנון מרחב חיפוש, והתמודדות עם סביבות דינמיות. מחקרים כגון "Q-learning for mobile robot navigation" (Kormushev et al., 2011) מציינים כיצד Q-learning שימש כדי לאפשר לרובוטים ללמוד להתמודד עם משימות ניווט בצורה אוטונומית.

2. **Deep Q Networks (DQN)**:

- **סקירה ספרותית** DQN: הוא שדרוג של Q-learning בו נעשה שימוש **ברשתות נוירוניות** עמוקות לצורך חיזוי ערכים (Q-values) של מצבים שונים DQN. מאפשר ללמוד ולהתמודד עם בעיות גדולות ומורכבות יותר, כמו בעיות ניווט במצבים שבהם יש חפיפות בין פעולות שונות ומצבים רבים ומגוונים.
- **מאמרים רלוונטיים** :המאמר של Mnih et al. (2015), "Human-level control through deep reinforcement learning", היה ציון דרך בתחום זה, שבו הוצג שימוש ב DQN-לשחק משחקים בצורה שדומה לאדם. מאמרים נוספים כמו "Solving robotic navigation with DQN" (Kurutach et al., 2018) מראים את השימוש המוצלח של DQN בבעיות ניווט רובוטי בסביבות שונות.

1.תכנון מסלול רובוטי

תכנון מסלול רובוטי הוא בעיה מרכזית בתחום הבינה המלאכותית והנדסת הרובוטיקה. הבעיה מתמקדת במציאת הדרך היעילה ביותר עבור רובוט לעבור ממיקום התחלתי למיקום יעד תוך התחשבות באילוצים כגון מכשולים, סביבות דינמיות וצריכת אנרגיה.

הרובוט פועל בסביבה שבה המידע המלא על המבנה שלה או על מצביה אינו תמיד זמין מראש. כך, יש להתמודד עם מצבים שבהם על הרובוט "ללמוד" את סביבתו תוך כדי תנועה ולהסתגל לשינויים בזמן אמת. היכולת לתכנן מסלול באופן אופטימלי היא קריטית עבור יישומים רבים, כגון רובוטים תעשייתיים, כלי רכב אוטונומיים ,רובוטי שירות ורחפניים אוטומטיים צבאיים.

האתגרים העיקריים בתכנון מסלול רובוטי:

1. **מכשולים בסביבה :**הסביבה שבה הרובוט פועל עשויה להכיל מכשולים פיזיים שאינם ידועים מראש או משתנים עם הזמן, דבר המאלץ את הרובוט להתאים את מסלולו בזמן אמת.
2. **אי ודאות :**מידע חלקי או שגוי על הסביבה יכול להוביל להחלטות שגויות, ולכן יש צורך במנגנונים שיאפשרו תיקון טעויות ושיפור לאורך זמן.
3. **אופטימיזציה :**מציאת מסלול אינה מספיקה – יש לשאוף למקסם את היעילות מבחינת משאבים, כגון זמן, אנרגיה או מספר פעולות.
4. **דינמיות :**בסביבות משתנות, הרובוט חייב להיות מסוגל להגיב במהירות לשינויים בלתי צפויים, כגון הופעת מכשולים חדשים או שינוי במיקום היעד.

במסגרת הפרויקט, אבחן את הבעיה דרך שתי גישות עיקריות:

1. **Q-learning** פתרון שבו הרובוט לומד מדינמיקת הסביבה ומבצע התאמות על סמך תגמולים, ללא צורך במודל מוקדם של הסביבה .
2. **Deep Q Networks (DQN)** שילוב של למידת חיזוק ורשתות נוירוניות עמוקות, המאפשר התמודדות עם סביבות מורכבות וגדולות במיוחד.

2. בחירת הגישות Q-learning ו-DQN לפתרון הבעיה

בחרתי בלמידת חיזוק כגישת פתרון עיקרית לבעיה של תכנון מסלול רובוטי מתוך סקרנות להבין כיצד תנועה אוטומטית של עצמים במרחב יכולה ללמוד ולהתפתח תוך כדי אינטראקציה עם הסביבה. בחרתי בשני האלגוריתמים הללו כי הם מספקים את הכלים להבין איך מכונה לומדת מתוך התנסות עם הסביבה.

ב **Q-learning**-הלמידה מתבצעת על ידי עדכון ערכים עבור כל פעולה שנעשית בסביבה, כאשר כל ערך מציין את הערך הצפוי של פעולה מסוימת במצב נתון. כלומר, המכונה לומדת אילו פעולות מביאות לתגמול הגבוה ביותר, ובכך בונה מדיניות פעולה אופטימלית. המידע הזה מאפשר להבין כיצד מכונה יכולה ללמוד בסביבה חדשה, ללא צורך במידע מראש. למעשה, הלמידה מתבצעת באמצעות ניסוי וטעיה, כלומר מתוך התגמולים והענישות שהמכונה מקבלת בעקבות פעולותיה.

ב- **Neural Networks** ובמיוחד ב DQN המכונה לא רק לומדת מהניסיון שלה, אלא עושה זאת באמצעות רשתות נוירונים עמוקות, שיכולות לעבד כמויות גדולות של נתונים ולהתאים את הפעולות בצורה גמישה יותר **DQN** .מהווה שדרוג לQ-learning, שכן הוא פותר בעיות של הכללת נתונים על ידי שימוש במודלים מתקדמים של רשתות נוירונים, שמאפשרים למכונה להתאים את המדיניות בצורה אופטימלית בסביבות דינמיות ומורכבות. המידע שמספק DQN הוא איך ניתן להכליל וללמוד מתוך סביבות אלו, שבהן יש הרבה יותר מצבים ופעולות.

Q-learning

יתרונות:

1. **פשטות**:
Q-learningהוא אלגוריתם פשוט יחסית להבנה וליישום. הוא מאפשר לרובוט ללמוד מתוך אינטראקציה עם הסביבה על ידי עדכון ערכים עבור כל פעולה בסביבה. כל פעולה מקבלת ערך צפוי, והרובוט לומד אילו פעולות מובילות לתוצאה הטובה ביותר.
2. **למידה מתוך ניסוי וטעיה**:
האלגוריתם לא דורש מידע מוקדם על הסביבה, ומאפשר לרובוט ללמוד ממעשיו והתגובות שהוא מקבל. עבור סביבות בהן מידע על המרחב אינו זמין מראש, זהו יתרון משמעותי.
3. **תאימות למצבים דינמיים בסביבות פשוטות**:
Q-learningמתאים מאוד לסביבות שבהן השינויים אינם קיצוניים מדי, וניתן לעדכן את המדיניות באופן רציף ויעיל תוך כדי תנועה.

חסרונות:

4. **סיבוכיות במרחבים גדולים**:
כאשר מדובר במרחבים דו-ממדיים גדולים או במערכות שבהן יש הרבה מצבים ופעולות, הטבלה של ערכי ה Q-עשויה להיות לא מעשית. בהגדרה Q-learning ,שומר טבלה שבה נשמר ערך Q עבור כל צירוף של מצב ופעולה, דבר שעשוי להפוך את האלגוריתם לכבד מאוד מבחינת זיכרון ועיבוד.
5. **קושי בהכללה**:
Q-learningמתקשה בהכללה כאשר יש צורך ללמוד מתוך כמות עצומה של נתונים או מצבים. הוא מבצע חיפוש על פי ערכים בודדים של Q עבור כל מצב, ולכן יכול להיות לא יעיל במצבים עם מגוון רחב של אפשרויות פעולה.

בעיה בה ניתן להשתמש בQ-learning:

תכנון מסלול רובוטי בסביבה דו-ממדית עם מכשולים סטטיים:
Q-learningהוא פתרון טוב כאשר יש סביבה דו-ממדית עם מכשולים מוכרים מראש. אם המכשולים ידועים והסביבה אינה משתנה באופן דינמי במהלך הזמן Q-learning ,יכול ללמוד לבצע אופטימיזציה של מסלול תוך כמה פעמים של חיפוש והתנסות. זו גישה טובה כשיש מידע בסיסי לגבי המכשולים וללא שינויים מהירים או לא צפויים בסביבה.

Deep Q Networks (DQN)

יתרונות:

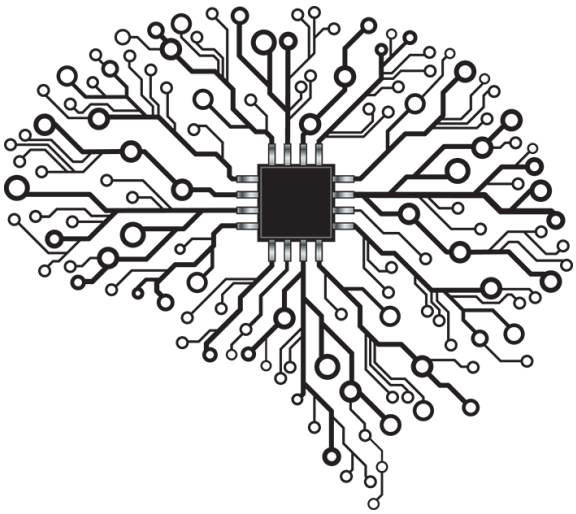
1. **יכולת הכללה גבוהה :**רשתות נוירונים מסוגלות ללמוד מתוך כמות גדולה של נתונים ולזהות קשרים בין פריטים שונים. הן לומדות להכליל מתוך הדוגמאות ולבצע תחזיות גם כאשר נתונים חדשים מגיעים .
2. **יכולת התאמה לסביבות דינמיות :**רשתות נוירונים יכולות לעדכן את הידע שלהן ולהגיב לשינויים בסביבה, הרשת יכולה לעדכן את פעולתה ולספק תחזית או תגובה מתאימה.

חסרונות:

1. **דרישות חישוביות גבוהות :**רשתות נוירונים דורשות משאבים חישוביים משמעותיים, במיוחד עבור רשתות גדולות ומורכבות. האימון דורש שימוש המון זיכרון, ולפעמים זמן חישוב ארוך מאוד.
2. **קושי בהגדרת פרמטרים :**האימון של רשת נוירונים דורש הגדרה של פרמטרים כמו מספר השכבות, מספר הנוירונים בכל שכבה, שיעור הלמידה ועוד. פרמטרים אלו דורשים ניסוי וטעיה, ולפעמים קשה למצוא את הקומבינציה האופטימלית.
3.

בעיה בה ניתן להשתמש ברשתות נוירונים:

תכנון מסלול רובוטי והימנעות ממכשולים :רשתות נוירונים יכולות לשמש בצורה מצוינת עבור בעיות של תכנון מסלול רובוטי, במיוחד כאשר מדובר בסביבות דינמיות עם מכשולים משתנים. בעיה כזו דורשת יכולת לזהות את המכשולים בסביבה בזמן אמת ולהגיב להם בצורה חכמה. הרשתות הנוירוניות, יכולות לנתח את המידע של הרובוט ולהבין את מבנה הסביבה. הן מסוגלות לעזור לרובוט ללמוד את הדרך האופטימלית להתנייד תוך הימנעות ממכשולים, ולתכנן מסלול במצבים משתנים ודינמיים שבהם יש הרבה משתנים ומידע לא ברור מראש.



3.סקירת המערכות

בפרויקט שלי שילבתי שתי גישות ללמידת רובוט לנווט בסביבה עם מכשולים:

1. **מערכת מבוססת Q-Learning**: את הקוד הבסיסי עבור מערכת זו מצאתי ב GitHub וביצעתי שינויים והתאמות כך שיתאימו לדרישות הפרויקט. שינויים אלו כללו את תוספת המפה, חיפוש המסלול ומציאת המסלול האופטימלי. בנוסף, השינויים שביצעתי כללו התאמות בתכנון המסלול הרובוטי כך שהמערכת תוכל להנחות את הרובוט במסלול אופטימלי. (reference)
2. **מערכת מבוססת רשת נוירונים**: גם קוד בסיסי עבור מערכת זו מצאתי ב GitHub, וביצעתי בו ההתאמות דומות כך שיתאים לדרישות הפרויקט. המערכת עושה שימוש ברשת נוירונים לפתור את אותה הבעיה. (reference)

מערכת הQ-Learning

שלב 1: יצירת הסביבה והגדרות מיקום.

- קובעים **רשת בגודל מוגדר** שמייצגת את העולם של הרובוט.
- מגדירים **נקודת התחלה ונקודת יעד**.
- מגדירים **מכשולים** עם ערך תגמול שלילי חזק, כדי לגרום לרובוט להימנע מהם.
- לכל צעד רגיל הרובוט מקבל **תגמול שלילי קל** כדי לעודד אותו למצוא מסלול מהיר.
- כשהרובוט מגיע ליעד , הוא מקבל **תגמול חיובי גדול** , כך שהוא ילמד לשאוף להגיע לשם
- יוצרים טבלה בגודל ועבור כל משבצת 4 ערכים. ערך לכל כיוון אפשרי

(up, down, left, right)

```
# קביעת פרמטרים של הסביבה
GRID_SIZE = 10  # גודל הרשת (10x10)
START = (0, 0)  # מיקום התחלתי
GOAL = (9, 9)  # מיקום היעד
ACTIONS = ['up', 'down', 'left', 'right']  # פעולות אפשריות
```

שלב 2: יצירת טבלת Q

- כל ערך Q מאותחל ל-0, כי הרובוט עדיין לא למד כלום.

```
# יצירת טבלת Q
Q_table = [[[0 for _ in range(len(ACTIONS))] for _ in range(GRID_SIZE)] for _ in range(GRID_SIZE)]
```

- במהלך הלמידה, הערכים בטבלה יתעדכנו כך שהרובוט ילמד איזו פעולה עדיפה בכל משבצת.

```
# עדכון ערך ה-Q לפי נוסחת עדכון Q-Learning
Q_table[position[0]][position[1]][action_index] += alpha * (
        reward + gamma * best_next_Q - Q_table[position[0]][position[1]][action_index])
```

שלב 3: יישום אלגוריתם Q-Learning

- הרובוט **עובר מספר פרקים מוגדר מראש (Episodes)** בהם הוא מנסה להגיע ליעד.
- בכל פרק הרובוט מתחיל מחדש ומבצע צעדים עד שהוא מגיע ליעד.
- בכל צעד הרובוט מחליט איזו פעולה לבצע לפי שתי אסטרטגיות:

1. **חקר – (Exploration)** לפעמים הרובוט יבחר צעד אקראי כדי לבדוק מסלולים חדשים.
2. **ניצול – (Exploitation)** הרובוט יעדיף לבחור את הצעד שמוביל לערך Q הכי גבוה.

לאחר כל צעד, הרובוט **מעודכן את טבלת Q** לפי הנוסחה:

- $Q(s, a) = Q(s, a) + \alpha * [R + \gamma * \max(Q(s', a')) - Q(s, a)]$
- alpha (שיעור למידה) – קובע כמה מהר הרובוט מעדכן את הידע שלו.
- gamma (מקדם הנחה) – קובע עד כמה הרובוט מתחשב בתגמולים עתידיים.
- R התגמול שהרובוט מקבל בצעד הנוכחי.
- Q של s ו-a, הערך הכי טוב של Q מהמשבצת הבאה.

```python
alpha = 0.5
gamma = 0.9
epsilon = 0.1

for episode in range(1000):  # מספר פרקים לאימון
    position = START  # אתחול למיקום ההתחלתי של הרובוט

    while position != GOAL:  # ריצה עד הגעה ליעד
        if random.uniform(0, 1) < epsilon:  # (exploitation) או ניצול הידע הקיים (exploration) החלטה אם לבצע חיפוש אקראי
            action = random.choice(ACTIONS)  # אם בחרנו חיפוש אקראי, בוחרים פעולה באופן אקראי מתוך פעולות אפשריות
        else:
            action = ACTIONS[max(range(len(ACTIONS)), key=lambda i: Q_table[position[0]][position[1]][i])]

        next_position = get_next_position(position, action)

        if rewards[next_position[0]][next_position[1]] == -100:
            continue  # אם יש מכשול, החזרה להתחלת הלולאה וניסיון פעולה אחרת

        reward = rewards[next_position[0]][next_position[1]]  # קבלת הערך של תגמול מהמיקום הבא
        best_next_Q = max(Q_table[next_position[0]][next_position[1]])  # מציאת הערך המקסימלי ב-Q-Table של המיקום הבא
        action_index = ACTIONS.index(action)  # זיהוי האינדקס של הפעולה הנבחרת

        Q_table[position[0]][position[1]][action_index] += alpha * (
                reward + gamma * best_next_Q - Q_table[position[0]][position[1]][action_index])

        print(f" :ערך Q חדש {Q_table[position[0]][position[1]][action_index]}")
        print("")

        position = next_position  # עדכון המיקום כדי להמשיך בפרק הבא
```

שלב 4: מציאת המסלול הטוב ביותר

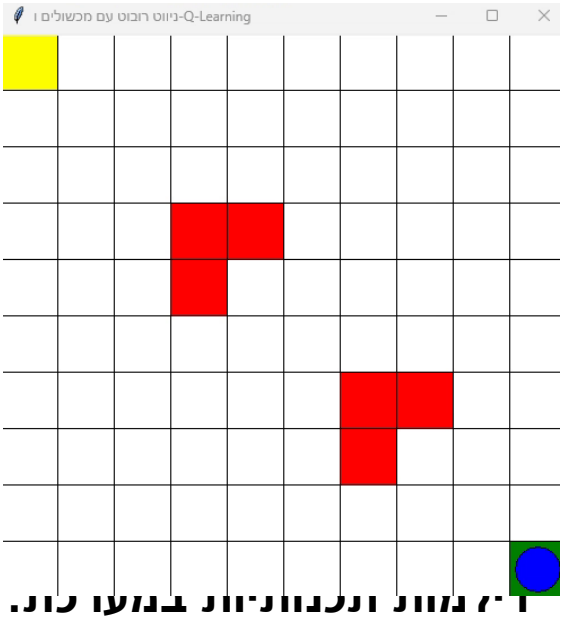
- לאחר שהרובוט למד, ניתן להשתמש בטבלת Q כדי למצוא את המסלול הקצר ביותר מההתחלה ליעד.
- בכל משבצת, הרובוט בוחר את הפעולה עם ערך Q הגבוה ביותר.

```
# מציאת המסלול הטוב ביותר
def find_path(start, goal):
    path = [start]
    position = start
    while position != goal:
        action = ACTIONS[max(range(len(ACTIONS)), key=lambda i: Q_table[position[0]][position[1]][i])]
        position = get_next_position(position, action)
        path.append(position)
    return path
```

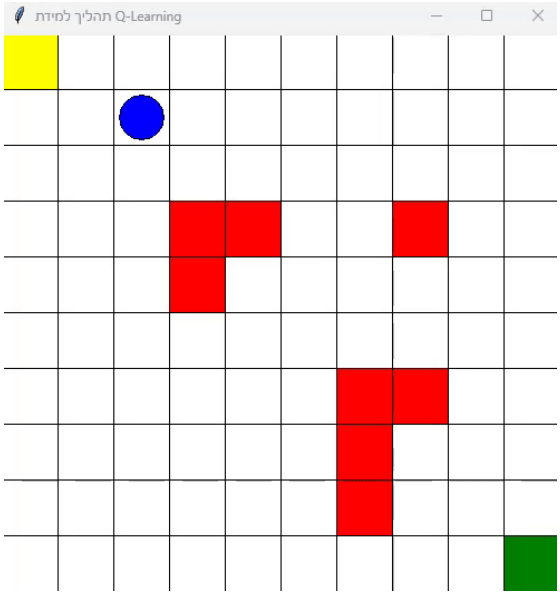
שלב 5: ממשק גרפי (GUI) להצגת המסלול

- נוצר חלון גרפי עם רשת של משבצות.
- הרובוט מיוצג כעיגול כחול, והיעד כמשבצת ירוקה.
- הרובוט זז באנימציה לפי המסלול שלמד, מתחמק ממכשולים המיוצגים במשבצת אדומה, ועובר צעד אחרי צעד מהמיקום ההתחלתי עד ההגעה אל היעד.

הצגת מסלול אופטימלי



הצגת תהליך הלמידה



הצגת הממשק הגרפי מול ביצועים

הייתה לי דילמה האם להמשיך להציג את הממשק הגרפי בזמן הריצה של האלגוריתם, או להפסיק את הצגתו על מנת לשפר את הביצועים. הצגת הממשק הגרפי דרשה זמן חישוב נוסף שהיה עלול להאט את תהליך הלמידה של הרובוט, במיוחד במהלך אימוני הQ-Learning.

במקום להציג את הממשק הגרפי בזמן אמת, החלטתי לעבור להדפסות בתוך המערכת כדי לעקוב אחרי התקדמות הרובוט בכל פרק. כך הצלחתי להאיץ את תהליך הלמידה ולבצע ניסויים מהר יותר. רק לאחר שהייתי מרוצה מהתוצאה הסופית של האלגוריתם, חזרתי להציג את הממשק הגרפי על מנת להדגים את המסלול האופטימלי שנמצא.

דילמה בבחירת פרמטרים ב Q-Learning

אחת הדילמות שהתמודדתי איתן הייתה בבחירת הפרמטרים המתאימים למערכת . לא הייתי בטוח באילו ערכים עליי לבחור עבור הפרמטרים השונים (alpha, gamma, epsilon),והשפעתם על ביצועי המערכת. מצד אחד, פרמטרים כמו epsilon ו-alpha משפיעים ישירות על הדרך בה הרובוט בוחר פעולות - חיפוש אקראי מול ניצול פעולות שנלמדו. השפעתם על ביצועי המערכת לא הייתה ברורה בהתחלה, ולא הייתי בטוח איזה שילוב ייתן תוצאה אופטימלית.

כיצד להפעיל את המערכת

1. **הורדת הקובץ** :הורד את הקובץ המכיל את הקוד המלא של המערכת למחשב (קובץ יחיד).
2. **פתיחת הקובץ** :פתח את הקובץ בפיתון.
3. **הפעלת הקוד** :אחרי פתיחת הקובץ, נדרש רק להריץ.

]

הסבר על מערכת Neural network(DQN)

חלק 1: יצירת הסביבה

- מחלקת GridWorld זו סביבה דמוי רשת בגודל מוגדר מראש שבה הרובוט מתחיל בנקודה
- פונקציה reset מאתחלת את מיקום הרובוט ומחזירה את המיקום ההתחלתי.
- פונקציה step: מקבלת את הפעולה שנבחרה (למעלה, למטה, שמאלה, ימינה) ומעדכנת את המיקום. מחזירה את המיקום החדש, את הציון ,וסטטוס סיום המשימה .

```python
# יצירת סביבה דמוית גריד
class GridWorld:  1 usage
    def __init__(self, size=10):
        self.size = size
        self.reset()

    def reset(self):  2 usages
        self.pos = [0, 0]  # מיקום התחלתי
        return self.pos

    def step(self, action):  1 usage
        x, y = self.pos
        if action == 0:  # למעלה
            x = max(0, x - 1)
        elif action == 1:  # למטה
            x = min(self.size - 1, x + 1)
        elif action == 2:  # שמאלה
            y = max(0, y - 1)
        elif action == 3:  # ימינה
            y = min(self.size - 1, y + 1)

        self.pos = [x, y]

        reward = -1
        done = False
        if self.pos == [self.size - 1, self.size - 1]:  # הגיע ליעד
            reward = 100
            done = True

        return self.pos, reward, done
```

חלק 2: יצירת רשת נוירונים(DQN)

```
# (DQN) רשת נוירונים
class DQN(nn.Module):  2 usages
    def __init__(self, input_dim, output_dim):
        super(DQN, self).__init__()
        self.fc = nn.Sequential(
            nn.Linear(input_dim,  out_features: 128), # שכבת קלט עם 128 נוירונים
            nn.ReLU(),# פונקציית אקטיבציה
            nn.Linear( in_features: 128,  out_features: 64), # שכבת חבילה של 64 נוירונים
            nn.ReLU(),# פונקציית אקטיבציה נוספת
            nn.Linear( in_features: 64, output_dim)  # (output_dim) שכבת פלט עם מספר פעולות האפשריות
        )

    def forward(self, x):
        return self.fc(x)# מחזיר את תוצאת השכבות הנוירוניות
```

א. פונקציה __init__(self, input_dim, output_dim)

הפונקציה __init__ מאתחלת את הרשת נוירונים ומגדירה את השכבות השונות שלה.

- **input_dim** גודל הקלט של הרשת. במקרה זה input_dim, הוא 2, כי יש שני משתנים המיקום של הרובוט ברשת (Y,X).
- **output_dim** מספר הפעולות האפשריות שהרובוט יכול לבצע. במקרה הזה, יש 4 פעולות אפשריות:
  - למעלה (0), למטה (1), שמאלה (2), ימינה (3).

ב. שכבות רשת הנוירונים

רשת הנוירונים בנויה ממספר שכבות

- **השכבה הראשונה**: מקבלת את הקלט, המיקום של הרובוט בגריד ומפיקה 128 נוירונים.
- **השכבה השנייה**: מפיקה 64 נוירונים.
- **השכבה האחרונה**: שכבת הפלט שמחזירה את הערכים של כל פעולה אפשרית

(למעלה, למטה, שמאלה וימינה). כל ערך כזה מייצג את הQ-value של פעולה מסוימת.

ג. פונקציה forward(self, x)

הפונקציה forward מבצעת חישוב של תוצאות הרשת. היא מעבירה את הקלט דרך השכבות המוגדרות ומחזירה את התוצאה הסופית הQ-values עבור כל פעולה אפשרית. ה Q-values משמשים להערכת הכדאיות של ביצוע כל פעולה במצב נתון.

בס"ד

3. הגדרת פרמטרים של האלגוריתם

- **סוג המידע:**
  - state_dim = 2 (x, y).
  - action_dim = 4 למעלה, למטה, שמאלה, ימינה.
- **פרמטרים של למידת חיזוק:**
  - learning_rate לימוד של הרשת.
  - gamma הנחות עתידית.
  - epsilon הסיכוי לבחור בחירה אקראית
  - epsilon_decay הקטנה הדרגתית של epsilon.
  - epsilon_min המינימום של epsilon.
  - batch_size גודל החבילות לאימון.
  - max_memory גודל הזיכרון.

```
# הגדרות
env = GridWorld()
state_dim = 2  # מיקום (x,y)
action_dim = 4  # 4 כיוונים
lr = 0.001
gamma = 0.9
epsilon = 1.0
epsilon_decay = 0.995
epsilon_min = 0.1
batch_size = 32
memory = []
max_memory = 1000
```

4. הגדרת האופטימיזציה:

- משתמשים ב Adam optimizer-כדי לעדכן את המשקלים של הרשת נוירונים, כלומר לשפר את הרשת.
- משתמשים ב MSE Loss-(ממוצע ריבועים) כדי למדוד את השגיאות של הרשת, כלומר כמה הרשת טעתה, ולמנוע טעויות בעתיד.

Adam עוזר לשפר את הרשת ו MSE-מודד כמה הרשת טעתה

.

```
optimizer = optim.Adam(q_network.parameters(), lr=lr)
loss_fn = nn.MSELoss()
```

5. הגדרת בחירת פעולה

- **פונקציה choose_action(state)**
  - אם random.random() < epsilon, מבוצעת פעולה אקראית (כדי לחקור את הסביבה).
  - אחרת, רשת הנוירונים בוחרת את הפעולה עם הQ-value הגבוה ביותר.

```
# פונקציה לבחירת פעולה
def choose_action(state):  1 usage
    if random.random() < epsilon:
        return random.randint( a: 0, action_dim - 1)
    state = torch.FloatTensor(state).to(device)
    with torch.no_grad():
        return torch.argmax(q_network(state)).item()
```

6. התחלת אימון

- ביצוע אימון על פי שלבים.
- **לכל פרק:**

1. אתחול הסביבה.state = env.reset()
2. חישוב reward מצטבר.
3. עדכון המיקום של הרובוט בהתאם לפעולה שנבחרה.
4. הרובוט בוחר פעולה ע"י **choose_action(state)** .
5. עדכון המיקום אחרי ביצוע הפעולה
6. שמירת התוצאה בזיכרון.
7. חישוב **Q-values** ומדידת הפסד עבור כל פעולה.
8. עדכון **Q-values** ברשת הנוירונים.
9. הפחתת (ε) epsilon באופן הדרגתי על מנת לאזן בין חקר לחexploitation .

7. אימון באמצעות דגימות אקראיות

- אם יש מספיק דוגמאות בזיכרון

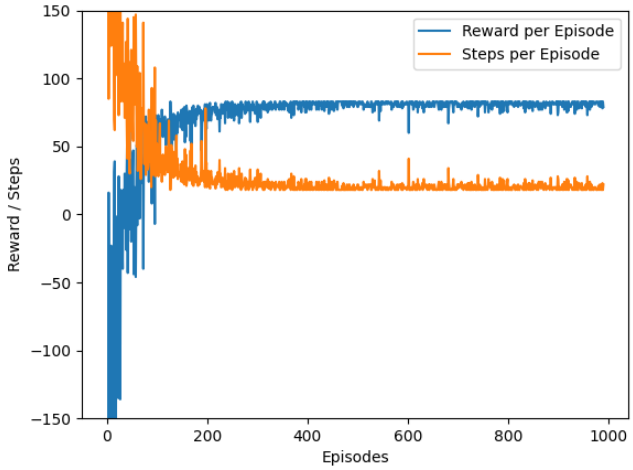
1. קח דגימה אקראית מהזיכרון.
2. עדכון ה Q-values-לכל דוגמה.
3. עדכון המשקלים של הרשת באמצעות חישוב ההפסד.

```
# אימון על דוגמאות רנדומיות
if len(memory) > batch_size:
    batch = random.sample(memory, batch_size)
    states, actions, rewards, next_states, dones = zip(*batch)

    states = torch.FloatTensor(states).to(device)
    actions = torch.LongTensor(actions).to(device)
    rewards = torch.FloatTensor(rewards).to(device)
    next_states = torch.FloatTensor(next_states).to(device)
    dones = torch.FloatTensor(dones).to(device)

    q_values = q_network(states).gather(1, actions.unsqueeze(1)).squeeze()
    next_q_values = q_network(next_states).max(1)[0]
    target_q_values = rewards + gamma * next_q_values * (1 - dones)

    loss = loss_fn(q_values, target_q_values.detach())
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

8. סיום האימון

- בסיום האימון, הצגת התוצאות הסופיות של האימון, כולל גרפים המתארים את השיפור לאורך הזמן. לדוגמה:

```
Best path: [[0, 0], [0, 1], [0, 2], [1, 2], [1, 3], [1, 4], [1, 5], [1, 6], [1, 7], [1, 8], [1, 7], [2, 7], [3, 7], [3, 8], [4, 8], [4, 9], [5, 9], [6, 9], [7, 9], [8, 9], [9, 9]]
Episode 900, Reward: 83, Epsilon: 0.1000, Steps: 18
Best path: [[0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [6, 0], [7, 0], [8, 0], [9, 0], [9, 1], [9, 2], [9, 3], [9, 4], [9, 5], [9, 6], [9, 7], [9, 8], [9, 9]]
Episode 910, Reward: 81, Epsilon: 0.1000, Steps: 20
Best path: [[0, 0], [1, 0], [0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [6, 0], [7, 0], [8, 0], [8, 1], [8, 2], [8, 3], [8, 4], [8, 5], [9, 5], [9, 6], [9, 7], [9, 8], [9, 9]]
Episode 920, Reward: 81, Epsilon: 0.1000, Steps: 20
Best path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [1, 5], [1, 6], [2, 6], [3, 6], [4, 6], [5, 6], [6, 6], [7, 6], [7, 7], [6, 7], [7, 7], [8, 7], [8, 8], [9, 8], [9, 9]]
Episode 930, Reward: 77, Epsilon: 0.1000, Steps: 24
Best path: [[0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [5, 1], [5, 2], [5, 3], [5, 4], [5, 5], [5, 4], [5, 5], [6, 5], [6, 4], [6, 5], [6, 6], [5, 6], [6, 6], [7, 6], [7, 7], [8, 7], [8, 8], [9, 8], [9, 9]]
Episode 940, Reward: 79, Epsilon: 0.1000, Steps: 22
Best path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [1, 6], [1, 7], [1, 8], [2, 8], [3, 8], [3, 9], [4, 9], [5, 9], [6, 9], [7, 9], [8, 9], [7, 9], [8, 9], [9, 9]]
Episode 950, Reward: 80, Epsilon: 0.1000, Steps: 21
Best path: [[0, 0], [1, 0], [2, 0], [3, 0], [3, 1], [3, 2], [3, 3], [3, 4], [3, 5], [3, 6], [4, 6], [5, 6], [6, 6], [6, 7], [7, 7], [7, 8], [7, 9], [6, 9], [7, 9], [7, 9], [8, 9], [9, 9]]
Episode 960, Reward: 82, Epsilon: 0.1000, Steps: 19
Best path: [[0, 0], [1, 0], [2, 0], [2, 0], [3, 0], [4, 0], [5, 0], [6, 0], [7, 0], [8, 0], [8, 1], [8, 2], [8, 3], [8, 4], [8, 5], [8, 6], [8, 7], [8, 8], [8, 9], [9, 9]]
Episode 970, Reward: 81, Epsilon: 0.1000, Steps: 20
Best path: [[0, 0], [1, 0], [2, 0], [3, 0], [3, 1], [4, 1], [5, 1], [5, 0], [6, 0], [7, 0], [8, 0], [8, 1], [8, 2], [8, 3], [8, 4], [8, 5], [8, 6], [8, 7], [8, 8], [9, 8], [9, 9]]
Episode 980, Reward: 79, Epsilon: 0.1000, Steps: 22
Best path: [[0, 0], [1, 0], [1, 1], [1, 2], [1, 3], [0, 3], [0, 4], [0, 5], [0, 6], [1, 6], [2, 6], [1, 6], [2, 6], [3, 6], [4, 6], [5, 6], [6, 6], [6, 7], [6, 8], [7, 8], [8, 8], [9, 8], [9, 9]]
Episode 990, Reward: 77, Epsilon: 0.1000, Steps: 24
Best path: [[0, 0], [0, 1], [0, 2], [1, 2], [2, 2], [3, 2], [2, 2], [2, 3], [3, 3], [4, 3], [4, 4], [4, 5], [5, 5], [6, 5], [6, 6], [7, 6], [7, 5], [7, 6], [6, 6], [7, 6], [7, 7], [8, 7], [8, 8], [9, 8], [9, 9]]
Training complete!
```



9. הערכת ביצועים

- לאחר סיום האימון, נעריך את ביצועי הרובוט על ידי ביצוע פרקים נוספים בהם הרובוט מבצע את הפעולות הטובות ביותר על פי רשת הנוירונים. לדוגמה:

```
Evaluating Performance...

Evaluation Episode 1: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [1, 2], [1, 3], [1, 4], [1, 5], [1, 6], [1, 7], [2, 7], [3, 7], [3, 8], [4, 8], [5, 8], [6, 8], [7, 8], [8, 8], [9, 8], [9, 9]]
Evaluation Episode 2: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [1, 2], [1, 3], [1, 4], [1, 5], [1, 6], [1, 7], [2, 7], [3, 7], [3, 8], [4, 8], [5, 8], [6, 8], [7, 8], [8, 8], [9, 8], [9, 9]]
Evaluation Episode 3: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [1, 2], [1, 3], [1, 4], [1, 5], [1, 6], [1, 7], [2, 7], [3, 7], [3, 8], [4, 8], [5, 8], [6, 8], [7, 8], [8, 8], [9, 8], [9, 9]]
Evaluation Episode 4: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [1, 2], [1, 3], [1, 4], [1, 5], [1, 6], [1, 7], [2, 7], [3, 7], [3, 8], [4, 8], [5, 8], [6, 8], [7, 8], [8, 8], [9, 8], [9, 9]]
Evaluation Episode 5: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [1, 2], [1, 3], [1, 4], [1, 5], [1, 6], [1, 7], [2, 7], [3, 7], [3, 8], [4, 8], [5, 8], [6, 8], [7, 8], [8, 8], [9, 8], [9, 9]]
Evaluation Episode 6: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [1, 2], [1, 3], [1, 4], [1, 5], [1, 6], [1, 7], [2, 7], [3, 7], [3, 8], [4, 8], [5, 8], [6, 8], [7, 8], [8, 8], [9, 8], [9, 9]]
Evaluation Episode 7: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [1, 2], [1, 3], [1, 4], [1, 5], [1, 6], [1, 7], [2, 7], [3, 7], [3, 8], [4, 8], [5, 8], [6, 8], [7, 8], [8, 8], [9, 8], [9, 9]]
Evaluation Episode 8: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [1, 2], [1, 3], [1, 4], [1, 5], [1, 6], [1, 7], [2, 7], [3, 7], [3, 8], [4, 8], [5, 8], [6, 8], [7, 8], [8, 8], [9, 8], [9, 9]]
Evaluation Episode 9: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [1, 2], [1, 3], [1, 4], [1, 5], [1, 6], [1, 7], [2, 7], [3, 7], [3, 8], [4, 8], [5, 8], [6, 8], [7, 8], [8, 8], [9, 8], [9, 9]]
Evaluation Episode 10: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [1, 2], [1, 3], [1, 4], [1, 5], [1, 6], [1, 7], [2, 7], [3, 7], [3, 8], [4, 8], [5, 8], [6, 8], [7, 8], [8, 8], [9, 8], [9, 9]]

Evaluation Results:
Average Reward: 83.00
Average Steps: 18.00
Success Rate: 100.00%
```

כיצד להפעיל את המערכת

1. **הורדת הקובץ**: הורד את הקובץ המכיל את הקוד המלא של המערכת למחשב (קובץ יחיד).
2. **פתיחת הקובץ**: פתח את הקובץ בפיתון.
3. **הפעלת הקוד**: אחרי פתיחת הקובץ, נדרש רק להריץ.

4. סיכום והשוואה בין האלגוריתמים

הרצת בעיה זהה על שני האלגוריתמים. הבעיה היא מפה בגודל 10X10 שבא הרובוט מנסה למצוא את היעד ב(9,9) תוך כדי הימנעות ממכשולים. נתחיל ברצת חלקית להבנה לעומק של ההסבר.

איך Q-learning עובד :

נחשב לפי הנוסחה $Q(s,a)=Q(s,a)+\alpha[R+\gamma \max Q(s',a')-Q(s,a)]$

צעד ראשון – מעבר מ(0,0) ל(0,1). ז"א נבחרה תנועה ימינה.

$Q(0,0,Right)=Q(0,0,Rig)+\alpha\times[R+\gamma\max(Q(0,1))-Q(0,0,Rig)]$

$Q(0,0,Right)=0+0.5\times[-1+0.9\times0-0]$

$Q(0,0,Right)=-0.5$

מצב	למעלה	למטה	שמאלה	ימינה
(0,0)	0	0	0	-0.5
(0,1)	0	0	0	0
(0,2)	0	0	0	0
(1,0)	0	0	0	0
(1,1)	0	0	0	0
(1,2)	0	0	0	0
(2,0)	0	0	0	0
(2,1)	0	0	0	0
(2,2)	0	0	0	0
...				

צעד שני – מעבר מ(0,1) ל(0,2). ז"א נבחרה תנועה ימינה.

$Q(0,1,Right)=Q(0,1,Rig)+\alpha\times[R+\gamma\max(Q(0,2))-Q(0,1,Rig)]$

$Q(0,1,Right)=0+0.5\times[-1+0.9\times0-0]$

$Q(0,0,Right)=-0.5$

מצב	למעלה	למטה	שמאלה	ימינה
(0,0)	0	0	0	-0.5
(0,1)	0	0	0	-0.5
(0,2)	0	0	0	0
(1,0)	0	0	0	0
(1,1)	0	0	0	0
(1,2)	0	0	0	0
(2,0)	0	0	0	0
(2,1)	0	0	0	0
(2,2)	0	0	0	0
...				

צעד שלישי – מעבר מ(0,2) ל(1,2). ז"א נבחרה תנועה למטה.

$Q(0,2,down)=Q(0,1,do)+\alpha\times[R+\gamma\max(Q(1,2))-Q(0,2,do)]$

$Q(0,1,Right)=0+0.5\times[-1+0.9\times0-0]$

$Q(0,0,Right)=-0.5$

מצב	למעלה	למטה	שמאלה	ימינה
(0,0)	0	0	0	-0.5
(0,1)	0	0	0	-0.5
(0,2)	0	-0.5	0	0
(1,0)	0	0	0	0
(1,1)	0	0	0	0
(1,2)	0	0	0	0
(2,0)	0	0	0	0
(2,1)	0	0	0	0
(2,2)	0	0	0	0
...				

הסבר הפרמטרים :

- **Q (s, a) -** ערך הפעולה במצב s.
- **α\alphaα** (שיעור הלמידה) – קובע כמה העדכון ישפיע על הערך הישן.
- **R**(גמול מידי) – התגמול שהתקבל מהפעולה.
- **γ\gamma** (מקדם ההנחה) – קובע חשיבות לגמולים עתידיים.
- **maxQ(s′,a′)** הערך המקסימלי הצפוי מהמצב הבא.

בחירת הצעד הבא:

הרובוט בוחר את הצעד הבא לפי ערכי ה Q-בטבלה ובעזרת פרמטר החקירה האקראית ε

1. **חקירה לעומת ניצול**
   - עם הסתברות ε הרובוט בוחר צעד אקראי מתוך הפעולות האפשריות. זה מאפשר לו לחקור מסלולים חדשים.
   - עם הסתברות ε פחות 1 הרובוט בוחר את הפעולה שמביאה לערך הQ הגבוה ביותר מהמצב הנוכחי, כלומר הפעולה שמניבה את הגמול המקסימלי הצפוי.
2. **שלבי הבחירה של הצעד הבא**
   - משיגים את כל הפעולות האפשריות מהמצב הנוכחי.
   - מגרילים מספר אקראי: אם הוא קטן מε מבצעים צעד אקראי.
   - אחרת, בוחרים את הפעולה עם ערך הQ הגבוה ביותר

חזרה אחת(Episode)

חזרה אחת במערכת מתייחס למסלול שלם שהרובוט עובר מהמצב ההתחלתי עד היעד, במהלך החזרה, הרובוט בוחר פעולות בכל מצב ואז מתבצע חישוב של הערכים בטבלה בהתאם לגמול שהושג.

כמה חזרות נדרשות

בשלב הראשון, הערכים בטבלה לא משקפים את הפתרון האופטימלי. ככל שהרובוט עובר יותר חזרות ומבצע יותר עדכונים, הערכים ב Q-table משתפרים והם מתקרבים לערכים האמיתיים של הציפיות לגמול המקסימלי. אין מספר קבוע של חזרות, ככל שמספר החזרות יהיה גבוה יותר הערכים בטבלה יהיו קרובים יותר לפתרון אופטימלי. ככל שמספר החזרות עולה, האלגוריתם לומד לאן ללכת כדי להשיג את התוצאה הטובה ביותר. מספר החזרות מוגדר מראש, אך ככל שיהיו יותר חזרות, תהליך הלמידה יתקרב לפתרון טוב יותר. כל חזרה משפרת את הערכים בטבלת הQ ,וככל שהמספר שלהן גבוה יותר, הערכים יהיו קרובים יותר לפתרון האופטימלי.

שיפור המסלול

במהלך החזרות, הרובוט לומד אילו פעולות מביאות לתוצאה טובה יותר (למשל, הגעה ליעד עם מינימום גמול שלילי). אחרי מספר חזרות, הרובוט מתחיל לבחור פעולות על פי הערכים המעודכנים בטבלה, שמצביעים על פעולות שמביאות לגמול גבוה יותר. המסלול משתפר כי הרובוט לומד את ההשפעה של כל פעולה ומעדכן את החלטותיו בהתאם, כך שהוא הולך ומגלה את המסלול היעיל ביותר עד ההגעה ליעד.

פתרון אופטימלי מתוך הטבלה

בסיום הלמידה, טבלת הQ מכילה את הגמול המקסימלי מכל מצב. כדי למצוא את המסלול האופטימלי, הרובוט יבחר את הפעולה שתוביל אותו למצב עם הערך הגבוה ביותר של Q, כלומר, הפעולה שתביא לו את הגמול המקסימלי. טבלת ה Q-עוזרת לרובוט לבחור תמיד את הפעולה הכי טובה במצב נתון, והמסלול האופטימלי הוא סדר הפעולות שמביא לגמול המקסימלי, עד שהרובוט מגיע ליעד.

מסלול חלקי אופטימלי אפשרי לאחר תהליך הלמידה.

- כאשר הרובוט נמצא במשבצת 0,0 הצעד עם הגמול הגבוה ביותר לפי טבלת הQ יהיה למטה ל1,0 או ימינה ל0,1 וכן הלאה.

מצב	למעלה	למטה	שמאלה	ימינה
(0,0)	62.17	70.19	62.17	70.19
(0,1)	70.19	79.10	62.17	79.10
(0,2)	79.10	89.00	70.19	79.10
(1,0)	30.27	79.10	50.78	59.02
(1,1)	52.40	76.37	34.45	89.00
(1,2)	79.10	100.00	79.10	89.00
(2,0)	34.03	39.01	65.93	89.00
(2,1)	56.68	44.25	58.43	100.00
(2,2)	0.00	0.00	0.00	0.00

השפעת הפרמטרים של Q-Learning על חיפוש מסלול

- α שיעור הלמידה – קובע כמה מהר האלגוריתם מעדכן את ערכי ה Q בכל צעד.
- γ מקדם ההנחה – קובע עד כמה האלגוריתם מתחשב בתגמולים עתידיים לעומת תגמול מידי.
- ε חקירה אקראית – קובע כמה מהצעדים יתבצעו באופן אקראי כדי לחקור מסלולים חדשים.

בס"ד

הרצאת האלגוריתם Q-learning על שלושה תסריטים עם השפעה שונה על חיפוש מסלול:

1. **פרמטרים אמינים שיובילו לתוצאה**

- alpha = 0.9 שיעור למידה גבוה מאפשר לרובוט ללמוד במהירות ולהתאים את התנהגותו בהתבסס על חוויות חדשות. הדבר מאפשר לו לשפר את החלטותיו תוך זמן קצר.
- gamma = 0.8 מקדם ההנחה משפיע על איך הרובוט מתחשב בתגמולים עתידיים. הערך של 0.8 מראה שהרובוט מתחשב בעתיד, אך גם נותן משקל גבוה לתגמולים מיידיים, מה שמוביל אותו לאזן בין חקר והנאה מיידית.
- epsilon = 0.1 מאפשר לרובוט לחקור 10% מהזמן פעולות אקראיות, מה שגורם לו לגלות אפשרויות חדשות שלא בהכרח היו משתלמות בהתחלה, אך עשויות להביא לתוצאות טובות יותר בטווח הארוך.

בנוסף הגדרתי **1000** חזרות לימוד. על מנת מאפשרת לרובוט לחוות מספר מספיק של מצבים וללמוד מהם.

```python
# קביעת פרמטרים של הסביבה
GRID_SIZE = 10  # גודל הרשת (10x10)
START = (0, 0)  # מיקום התחלתי
GOAL = (9, 9)  # מיקום היעד
ACTIONS = ['up', 'down', 'left', 'right']  # פעולות אפשריות

# יצירת טבלת תגמולים עם מכשולים
rewards = [[-1 for _ in range(GRID_SIZE)] for _ in range(GRID_SIZE)]  # כל תזוזה מקבלת תגמול שלילי

# מכשולים (ערכים של -100 כדי שלא ייבחרו)
obstacles = [(3, 3), (3, 4), (4, 3), (6, 6), (6, 7), (7, 6)]
for obs in obstacles:
    rewards[obs[0]][obs[1]] = -100

# היעד מקבל תגמול גבוה
rewards[GOAL[0]][GOAL[1]] = 100

# יצירת טבלת Q
Q_table = [[[0 for _ in range(len(ACTIONS))]
            for _ in range(GRID_SIZE)]
            for _ in range(GRID_SIZE)]

# פרמטרים של Q-Learning
alpha = 0.9  # שיעור למידה  משפיע על איך נעדכן את ה-Q-Table.
gamma = 0.8  # מקדם הנחה קובע עד כמה נחשיב את העתיד (הערכים העתידיים של פעולות).
epsilon = 0.1  # הסתברות לבחירת פעולה אקראית (exploration) # קובע כמה פעמים נחפש פעולות חדשות לעומת ניצול הפעולות הטובות ביותר לפי ה-Q-Table.
```

פלט:

```
| -3.11  -3.36  -3.15  -2.64| -2.64  -2.29  -3.40  -2.04| -2.05  -1.31  -3.20  -1.34| -2.24  -0.38  -2.87  -2.17| -2.24  -1.78  -1.70  -1.64| -2.24  -1.78  -2.17  -1.64| -2.24  -2.69  -2.17  -2.34| -2.24  -2.65  -2.40  -2.39| -2.24  -2.60  -2.68  -2.40| -2.24  -2.26  -2.63  -2.24 |
| -3.47  -2.95  -3.15  -2.95| -3.15  -1.34  -3.30  -2.44| -2.04  -0.38  -2.95  -0.38| -2.44   0.77  -1.31   0.49| -1.65   2.21  -1.80  -2.18| -2.18  -1.80  -1.80  -2.64| -2.30  -2.30  -2.40  -2.70| -2.63  -2.40  -2.60  -2.70| -2.68  -2.41  -2.19  -2.29| -2.24  -2.24  -2.69  -2.24 |
| -3.13  -2.44  -2.74  -2.44| -2.18  -1.80  -2.95  -0.38| -2.44  -1.00  -2.44   0.77| -1.80   0.00  -0.38   2.22| 0.28   0.00   0.77   4.02| -2.43   6.27   2.18  -2.35| -2.71   3.39  -1.80  -2.24| -2.71  -2.30  -2.36  -2.24| -1.79  -2.24  -2.73  -2.10| -2.63  -2.24  -2.17  -2.24 |
| -2.63  -2.44  -2.24  -1.80| -2.44  -1.80  -2.44  -1.00| -0.38  -1.00  -1.80   0.00| 0.00   0.00   0.00   0.00| 0.00   0.00   0.00   0.00| 4.01   9.09   0.00   2.91| -2.40  -1.80   6.27  -2.31| -2.40  10.80  -1.78  -2.24| -2.34  -1.71  -2.30  -2.24| -2.24  -2.24  -2.24  -2.24 |
| -2.41  -2.24  -2.24  -1.80| -1.80  -2.44  -2.44  -1.00| -1.00  -1.80  -1.80   0.00| 0.00   0.00   0.00   0.00| 0.00   0.93   0.00   9.09| -1.00  12.62   6.27  -1.80| -1.64  17.01  -1.55  -1.77| -1.64  22.44  -1.71  -1.64| -2.24  26.30  -1.78  -2.17| -2.24  -2.24  -2.24  -1.64 |
| -2.42  -2.24  -2.24  -2.10| -1.80  -2.26  -2.64  -1.80| -1.00  -2.44  -2.44  -1.00| 0.00  -1.80  -1.80  -1.80| 6.27  -1.78  -1.00  -1.64| 8.98   9.08   3.91  17.02| 12.42   0.00  12.61  22.53| 14.52   0.00  16.84  29.41| -2.24  38.01  22.29  -2.35| -1.71  -1.71  26.30  -2.24 |
| -2.24  -1.71  -2.24  -2.24| -2.40  -2.24  -2.24  -2.29| -1.80  -2.24  -2.24  -1.80| -1.00  -2.30  -2.36  -1.80| -1.64  -1.64  -1.70   6.73| 12.62  -1.00  -1.80   0.00| 0.00   0.00   0.00   0.00| 0.00   0.00   0.00   0.00| 28.26  48.76   0.00  23.65| -1.64  -1.64  37.62  -1.64 |
| -1.71  -1.64  -1.64  -1.64| -2.24  -2.24  -2.17  -2.24| -2.24  -1.78  -2.24  -2.29| -1.71  -1.71  -2.24  -1.79| -1.64  -1.65  -1.64   9.40| -1.00  25.10  -1.80   0.00| 0.00   0.00   0.00   0.00| 0.00  47.03   0.00  -1.80| 34.11  62.20  21.62  60.40| -1.65  79.00  -1.55  -1.64 |
| -1.64  -1.64  -0.99  -1.64| -2.24  -1.64  -1.64  -1.64| -2.24  -0.99  -1.64  -1.64| -2.24  -1.64  -1.64  -1.64| -1.64  -1.64  -1.64  -1.71| -1.00  -2.24  -1.55  37.29| 0.00  -1.80  -1.80  48.74| -0.99  -1.78  34.07  62.20| 48.75  78.92  48.75  79.00| 61.55  100.00  62.19  71.10
| -1.64  -0.90  -0.90  -0.90| -1.64  -0.90  -0.90  -0.90| -0.99  -0.90  -0.90  -0.90| -1.64  -1.64  -1.55  -1.64| -1.64  -1.64  -1.64  -1.64| -1.78  -1.64  -1.64  -1.64| -1.00  -1.64  -1.64  44.87| -1.65  -1.64  -1.55  76.76| -0.90  -0.90  -1.55  100.00| 0.00  0.00  0.00  0.00 |
מסלול אופטימלי:
(0, 0), (0, 1), (0, 2), (1, 2), (1, 3), (2, 3), (2, 4), (2, 5), (3, 5), (3, 6), (4, 6), (5, 6), (5, 7), (5, 8), (6, 8), (6, 9), (7, 9), (8, 9), (9, 9)
```

סיכום הרצה : הבחירות של הערכים הובילו לאיזון טוב בין חקר לבין ניצול, והגבירו את יכולת הלמידה של הרובוט. עם הזמן, הרובוט היה מסוגל למצוא מסלולים אופטימליים תוך שמירה על יכולת לחקור ולגלות פתרונות חדשים. חזרות הלימוד הרבות סייעו בייעול התהליך ושיפור היכולת של הרובוט לקבל החלטות.

בס"ד

2. הגדרת המפה בגודל 100X100 ומיקום המטרה בנקודה (99,99) עם 100 חזרות לימוד.

```
# קביעת פרמטרים של הסביבה
GRID_SIZE = 100  # גודל הרשת (10x10)
START = (0, 0)  # מיקום התחלתי
GOAL = (99, 99)  # מיקום היעד
ACTIONS = ['up', 'down', 'left', 'right']  # פעולות אפשריות
```

ייתכן שהקוד עדיין רץ ברגעים אלה ממש.

המפה הוגדרה בגודל 100 על 100, מה שמוביל לכמות גדולה מאוד של מצבים שצריך לעבור במהלך החיפוש. גודל המפה משפיע ישירות על זמן הריצה של האלגוריתם, שכן יש לקחת בחשבון את כל האפשרויות האפשריות בכל שלב, מה שמוביל לעומס חישובי גבוה.

למרות השימוש בפרמטרים שמסייעי מבחינת זמן הריצה, , הבעיה הפכה ל

```
מ-(8 ,6) בחרנו פעולה right -> מיקום הבא (9 ,6)
ערך Q קודם: -1.8
ערך Q חדש: -1.8

מ-(9 ,6) בחרנו פעולה left -> מיקום הבא (8 ,6)
ערך Q קודם: -1.0
ערך Q חדש: -1.0

מ-(8 ,6) בחרנו פעולה right -> מיקום הבא (9 ,6)
ערך Q קודם: -1.8
ערך Q חדש: -1.8

מ-(9 ,6) בחרנו פעולה left -> מיקום הבא (8 ,6)
```

3. מסלול לא אופטימלי והתנהגות לא צפויה.

```
# קביעת פרמטרים של הסביבה
GRID_SIZE = 10  # גודל הרשת (10x10)
START = (0, 0)  # מיקום התחלתי
GOAL = (9, 9)  # מיקום היעד
ACTIONS = ['up', 'down', 'left', 'right']  # פעולות אפשריות

# יצירת טבלת תגמולים עם מכשולים
rewards = [[-1 for _ in range(GRID_SIZE)] for _ in range(GRID_SIZE)]  # כל תזוזה מקבלת תגמול שלילי

# מכשולים (ערכים של -100 כדי שלא ייבחרו)
obstacles = [(3, 3), (3, 4), (4, 3), (6, 6), (6, 7), (7, 6)]
for obs in obstacles:
    rewards[obs[0]][obs[1]] = -100

# היעד מקבל תגמול גבוה
rewards[GOAL[0]][GOAL[1]] = 100

# יצירת טבלת Q
Q_table = [[[0 for _ in range(len(ACTIONS))]
            for _ in range(GRID_SIZE)]
            for _ in range(GRID_SIZE)]

# פרמטרים של Q-Learning
alpha = 0.9  # שיעור למידה  משפיע על איך נעדכן את ה-Q-Table.
gamma = 0.2  # מקדם הנחה קובע עד כמה נחשיב את העתיד (הערכים העתידיים של פעולות).
epsilon = 0.5  # קובע כמה פעמים נחפש פעולות חדשות לעומת ניצול הפעולות הטובות ביותר לפי ה-Q-Table.
```

- **alpha = 0.9** שיעור למידה גבוה
- **gamma = 0.2** מקדם הנחה נמוך
- **epsilon = 0.5** חיפוש אקראי גבוה
- **100חזרות לימוד**

1 alpha גבוה מדי

- **ההשפעה** :שיעור למידה גבוה מדי גורם לרובוט ללמוד מהר מדי על פעולות מיידיות מבלי לתת מספיק זמן לבחינת כל האפשרויות.
- **תוצאה** :הרובוט עשוי לבחור באופציות לא אופטימליות על בסיס תגמול מידי.

2 gamma נמוך מדי

- **ההשפעה** :מקדם הנחה נמוך יגרום לרובוט להעדיף תגמולים מיידיים על פני האפשרויות העתידיות. הוא יחשוב בעיקר על היתרונות המיידיים שהוא יכול להפיק מהפעולות הנוכחיות.
- **תוצאה** :הרובוט יפספס את הדרך הארוכה יותר אך האופטימלית, כי הוא לא יראה את היתרונות של פעולות שיביאו לתגמול עתידי גבוה יותר.

3 epsilon גבוה מדי

- **ההשפעה** :אחוז גבוה מדי של חיפוש אקראי (50% מהזמן) גורם לרובוט לבחור בפעולות אקראיות מבלי להסתמך על הלמידה הקודמת שלו. זה מונע ממנו להתמקד במסלול האופטימלי.
- **תוצאה** :רוב הזמן הרובוט יבצע פעולות אקראיות, במקום להתרכז במסלול שיכול להוביל אותו למטרה בצורה אופטימלית.

4 100 חזרות לימוד

- **ההשפעה** :המספר הקטן של חזרות לימוד לא מספיק כדי לאפשר לרובוט לחקור את כל הסביבה ולהתאים את ההתנהגות שלו בצורה אופטימלית. הוא עשוי ללמוד את המפה באופן רשלני.
- **תוצאה** :הרובוט לא מספיק ללמוד את כל האפשרויות ואת השפעות הפעולות השונות, מה שגורם לו לקבל החלטות לא נכונות במצבים מסוימים.

השילוב של ערכים לא אופטימליים עבור הפרמטרים השונים גורם לרובוט לפעול בצורה לא יציבה ולבחור במסלולים שאינם בהכרח הטובים ביותר. שיעור הלמידה הגבוה מוביל ללמידה חפוזה, מקדם ההנחה הנמוך מקשה על ראיית יתרונות ארוכי טווח, רמת החיפוש האקראי הגבוהה מונעת התמקדות במסלול אופטימלי, ומספר החזרות המצומצם אינו מאפשר רכישת ידע מספק. כדי לשפר את ביצועי הרובוט.

בס"ד

תובנות לגבי Q-Learning

זיכרון ועיבוד – שם המשחק

כמות החישובים הנדרשת למציאת מסלול אופטימלי היא עצומה ודורשת **משאבי זיכרון משמעותיים** .כל מצב במפה צריך להילקח בחשבון, וכל מעבר אפשרי בין המצבים מוסיף עוד שכבות של חישוב. ככל שהמפה גדולה ומורכבת יותר, כך כמות הזיכרון והעיבוד הנדרשת גדלה באופן אקספוננציאלי.

חשיבות הפרמטרים

כדי שהאלגוריתם ימצא מסלול אופטימלי ויתפקד באופן יציב, יש **לכוונן את הפרמטרים בקפידה**:

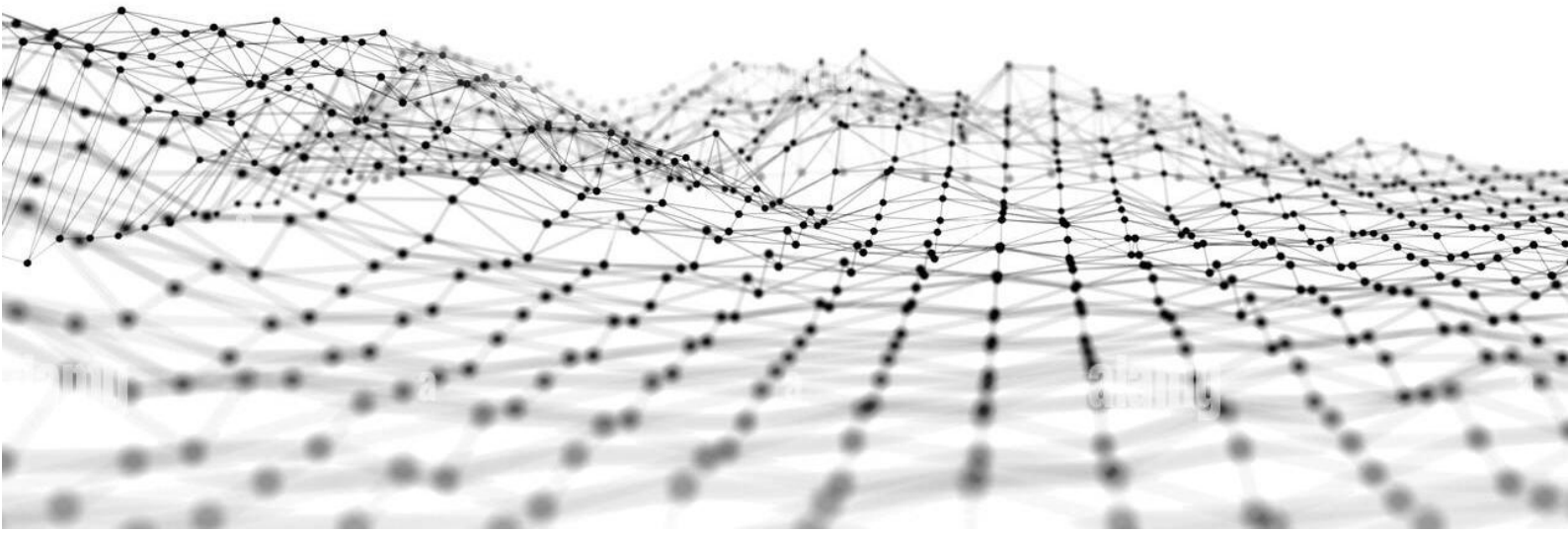
- **שיעור הלמידה :(α)** קובע עד כמה ערכי ה Q-מתעדכנים בכל שלב. ערך גבוה מדי יגרום לשינויים חדים ולא יציבים, וערך נמוך מדי יוביל ללמידה איטית.
- **מקדם ההנחה :(γ)** קובע את החשיבות של תגמולים עתידיים. ערך נמוך מדי יגרום לאלגוריתם להתמקד רק בטווח הקצר, בעוד שערך גבוה יעודד תכנון לטווח ארוך.
- **יחס חקירה-ניצול :(ε)** חייב להיות מאוזן. חקירה מוגזמת גורמת ללמידה אקראית, וניצול מוגזם גורם לאלגוריתם להיתקע במסלול לא אופטימלי.

השפעת גודל המפה ומורכבות הבעיה

במפות קטנות עם מסלול ברור Q-Learning ,מתכנס מהר יחסית. אבל ככל שהמפה גדלה ומתווספים מכשולים, מספר החזרות שצריך לבצע כדי להגיע לפתרון יציב עולה משמעותית. במפות מורכבות במיוחד, ייתכן ש Q-Learning-בלבד לא יספיק, ויהיה צורך בשיטות מתקדמות כמו **Deep Q-Learning**.

סיכום – התאמה נכונה של הפרמטרים היא קריטית

ללא תיאום נכון של הפרמטרים, האלגוריתם עלול להוביל למסלול **לא אופטימלי** או אפילו להתנהגות בלתי צפויה. לכן, אחד הלקחים המרכזיים הוא **להשקיע זמן בבחירת הפרמטרים הנכונים** ,להתאים אותם בהתאם לגודל המפה ולתנאי הבעיה, ולבצע בדיקות כדי לוודא שהאלגוריתם מתכנס לפתרון הרצוי.



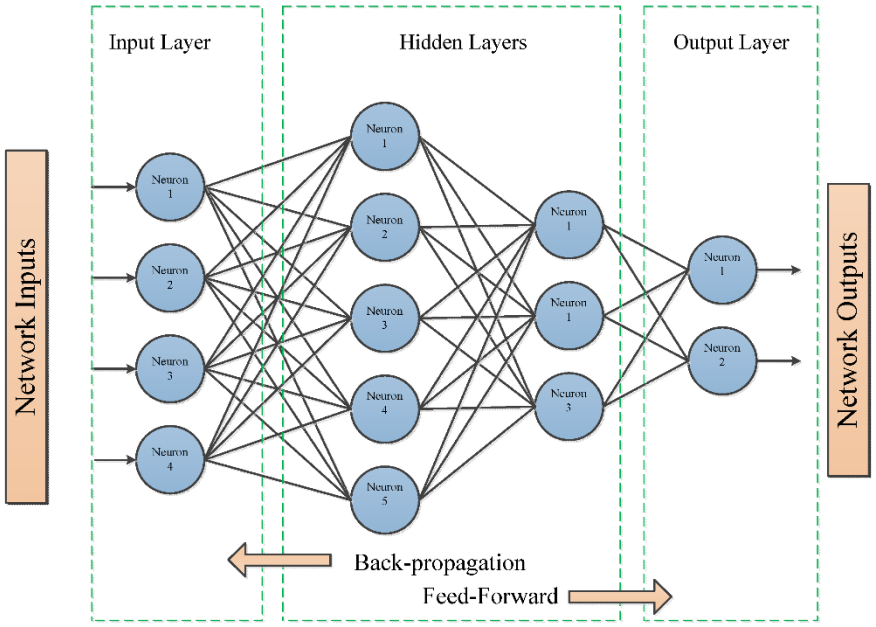
בס"ד

איך DQN עובד :

רשת הנוירונים במערכת שלי כוללת 3 שכבות.

- שכבת קלט שבא שתי נוירונים, המייצגים את המצב הנוכחי של הרובוט (x,y)
- שכבה מוסתרת אחת שהיא שיכבה עם 128 נוירונים.
- שכבת פלט שבא יש 4 נוירונים, אחד לכל פעולה אפשרית (למעלה, למטה, ימינה, שמאלה)

1. **הקלט – מצב הרובוט**:
   - הרובוט מקבל את המידע על המצב הנוכחי שלו, המיקום במישור, כלומר זוג קואורדינטות (x,y)
   - המידע הזה מועבר לשכבת הקלט של הרשת.
2. **שכבת קלט**:
   - בשכבת הקלט יש שני נוירונים, שמייצגים את המיקום הנוכחי של הרובוט במישור
   - המטרה של השכבה הזו היא להמיר את המידע (הקואורדינטות) לפורמט שניתן לעיבוד בשכבות הבאות.
3. **שכבה מוסתרת**:
   - בשכבה זו יש 128 נוירונים, שהיא שכבת עיבוד לא לינארית.
   - השכבה מנתחת את המידע שהתקבל משכבת הקלט ומבצעת חישובים פנימיים כדי למצוא תבניות ודפוסים במידע.
   - כל נוירון בשכבה מקבל את המידע מהשכבה הקודמת ומשתמש בפונקציות הפעלה כמו ReLUכדי למפות את המידע ולהפיק תוצאות שימושיות להחלטות עתידיות.
4. **שכבת פלט**:
   - בשכבה זו יש 4 נוירונים, שכל אחד מהם מייצג את הערך Q(s,a) של פעולה אפשרית בסיטואציה נתונה: למעלה, למטה, ימינה, שמאלה.
   - הערכים הללו מציינים את התגמול שהרובוט מצפה לקבל אם יבצע כל פעולה.



תהליך קבלת ההחלטות

ברגע שהרובוט מקבל את המידע אודות מצבו, הרשת מחשבת את הערכים Q(s,a) לכל פעולה אפשרית במצב זה:

1. לכל פעולה a במצב s, מחושב הערך Q(s,a) שמייצג את הצפי לתגמול שיתקבל אם הפעולה תבוצע.
2. הרובוט בוחר את הפעולה בעלת הערך Q(s,a) הגבוה ביותר.

לדוגמה, אם הערכים עבור כל פעולה הם:

- למעלה = 3.2
- למטה = 0.5
- ימינה = 2
- שמאלה = 1.3

אז הרובוט יבחר לפעול "למעלה" כי זהו הערך הגבוה ביותר.

עדכון הערכים – למידת חיזוק

לאחר שהרובוט בחר פעולה, הוא מקבל תגמול r על פי התוצאה של הפעולה. על פי התגמול הזה, ערך Q(s,a) מתעדכן כדי לשפר את החלטותיו בעתיד. תהליך זה נעשה באמצעות נוסחת ה-Q learning :

$$Q(s,a)=Q(s,a)+\alpha(r+\gamma\cdot a'\max Q(s',a')-Q(s,a)$$

כאשר:

- Q(s,a) הערך הנוכחי של פעולה a במצב s .
- r התגמול שהרובוט קיבל לאחר ביצוע הפעולה.
- קצב הדיפוי ,(discount factor) שמסביר את המשמעות של תגמולים עתידיים לעומת תגמולים מיידיים.
- $\max' Q(s',a')$ הערך הגבוה ביותר מכל פעולה אפשרית במצב הבא s המצב שהרובוט יגיע אליו לאחר ביצוע הפעולה.
- α\alphaα שיעור הלמידה ,שמסביר עד כמה מהר הערכים יתעדכנו ביחס לתגמול החדש.

עדכון הערכים

העדכון של הערכים Q(s,a) הוא קריטי כי הוא מאפשר לרובוט ללמוד מהניסיון ולטפל במצבים שבהם הוא קיבל תגמול נמוך או גבוה. הרובוט לא יודע מראש את התוצאה של כל פעולה, אבל הוא לומד תוך כדי הניסיון, ומבצע התאמות בקבלת ההחלטות בהמשך.

תהליך כללי של:DQN

1. **הקלט** :הרובוט מקבל את המיקום הנוכחי שלו (x,y)
2. **שכבות עיבוד** :המידע עובר דרך השכבות השונות של הרשת (קלט, מוסתרת, פלט), שמחשבות את הערכים Q(s,a) לכל פעולה אפשרית.
3. **החלטה** :הרובוט בוחר את הפעולה עם הערך הגבוה ביותר.
4. **ביצוע ותגמול** :הרובוט מבצע את הפעולה ומקבל תגמול.
5. **עדכון הערכים** :על פי התגמול, הערכים Q(s,a) מתעדכנים כדי לשפר את קבלת ההחלטות העתידית.
6. **למידה לאורך זמן** :הרובוט ממשיך ללמוד ולשפר את החלטותיו ככל שמצבים חדשים מופיעים.

הרצה 1, הרצה עם פרמטרים אמינים:
פרמטרים שהוכנסו למערכת

```
# הגדרות
env = GridWorld()
state_dim = 2
action_dim = 4
lr = 0.001
gamma = 0.9
epsilon = 1.0
epsilon_decay = 0.995
epsilon_min = 0.1
batch_size = 32
memory = []
max_memory = 1000
```

1. **Epsilon (ε):**
   - ערכו ההתחלתי היה גבוה אך במהלך האימון ירד בהדרגה עד 0.1.
   - מסמל את קצב החקירה – (Exploration Rate) הסיכוי לבחור פעולה אקראית במקום לבחור את הפעולה האופטימלית הידועה.
   - ערך נמוך בסוף האימון מעיד שהמודל מסתמך יותר על הידע שנצבר ולא על ניסיונות אקראיים.

```
# אימון
for episode in range(1000):
```

2. **מספר האפיזודות:(Episodes)**
   - האימון נמשך 1000 אפיזודות כלומר, 1000 ניסיונות שונים.
   - בכל אפיזודה הסוכן התחיל מנקודת ההתחלה וניסה למצוא את המסלול האופטימלי לסיום.
3. **תגמול:(Reward)**
   - בסוף האימון התגמול המתקבל היה 81 .
   - לאחר ההערכה, התגמול הממוצע עלה ל83 ,מה שמעיד על שיפור בהתנהגות הסוכן.
4. **מספר הצעדים:(Steps)**
   - בסוף האימון המסלול הטוב ביותר הושג ב 20-צעדים.
   - בהערכה, הסוכן שיפר את הביצועים והגיע ליעד ב 18-צעדים בלבד.

תוצאות האימון

- המסלול הטוב ביותר שנמצא בסוף האימון לפני ההערכה

```
Episode 960, Reward: 81, Epsilon: 0.1000, Steps: 20
Best path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [0, 7], [0, 8], [1, 8], [2, 8], [3, 8], [4, 8], [5, 8], [6, 8], [6, 7], [7, 7], [8, 7], [8, 8], [9, 8], [9, 9]]
Episode 970, Reward: 83, Epsilon: 0.1000, Steps: 18
Best path: [[0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [6, 0], [7, 0], [8, 0], [9, 0], [9, 1], [9, 2], [9, 3], [9, 4], [9, 5], [9, 6], [9, 7], [9, 8], [9, 9]]
Episode 980, Reward: 79, Epsilon: 0.1000, Steps: 22
Best path: [[0, 0], [0, 1], [1, 1], [1, 2], [1, 3], [1, 4], [1, 5], [0, 5], [0, 6], [0, 7], [0, 8], [0, 9], [0, 9], [0, 9], [1, 9], [2, 9], [3, 9], [4, 9], [5, 9], [6, 9], [7, 9], [8, 9], [9, 9]]
Episode 990, Reward: 81, Epsilon: 0.1000, Steps: 20
Best path: [[0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [5, 1], [6, 1], [7, 1], [8, 1], [9, 1], [9, 2], [9, 3], [9, 4], [9, 5], [9, 6], [9, 7], [9, 8], [8, 8], [9, 8], [9, 9]]
Training complete!
```

בס"ד

תוצאות ההערכה

לאחר סיום האימון, בוצעו 10אפיזודות הערכה ,והתקבלו התוצאות הבאות:

- תגמול ממוצע 83.
- מספר צעדים ממוצע 18 .
- שיעור הצלחה 100% כלומר, בכל ההרצות הסוכן הצליח להגיע ליעד.
- המסלול שהסוכן בחר בכל האפיזודות:

```
Evaluation Episode 1: Reward = 83, Steps = 18, Path: [[0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [6, 0], [7, 0], [8, 0], [9, 0], [9, 1], [9, 2], [9, 3], [9, 4], [9, 5], [9, 6], [9, 7], [9, 8], [9, 9]]
Evaluation Episode 2: Reward = 83, Steps = 18, Path: [[0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [6, 0], [7, 0], [8, 0], [9, 0], [9, 1], [9, 2], [9, 3], [9, 4], [9, 5], [9, 6], [9, 7], [9, 8], [9, 9]]
Evaluation Episode 3: Reward = 83, Steps = 18, Path: [[0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [6, 0], [7, 0], [8, 0], [9, 0], [9, 1], [9, 2], [9, 3], [9, 4], [9, 5], [9, 6], [9, 7], [9, 8], [9, 9]]
Evaluation Episode 4: Reward = 83, Steps = 18, Path: [[0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [6, 0], [7, 0], [8, 0], [9, 0], [9, 1], [9, 2], [9, 3], [9, 4], [9, 5], [9, 6], [9, 7], [9, 8], [9, 9]]
Evaluation Episode 5: Reward = 83, Steps = 18, Path: [[0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [6, 0], [7, 0], [8, 0], [9, 0], [9, 1], [9, 2], [9, 3], [9, 4], [9, 5], [9, 6], [9, 7], [9, 8], [9, 9]]
Evaluation Episode 6: Reward = 83, Steps = 18, Path: [[0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [6, 0], [7, 0], [8, 0], [9, 0], [9, 1], [9, 2], [9, 3], [9, 4], [9, 5], [9, 6], [9, 7], [9, 8], [9, 9]]
Evaluation Episode 7: Reward = 83, Steps = 18, Path: [[0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [6, 0], [7, 0], [8, 0], [9, 0], [9, 1], [9, 2], [9, 3], [9, 4], [9, 5], [9, 6], [9, 7], [9, 8], [9, 9]]
Evaluation Episode 8: Reward = 83, Steps = 18, Path: [[0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [6, 0], [7, 0], [8, 0], [9, 0], [9, 1], [9, 2], [9, 3], [9, 4], [9, 5], [9, 6], [9, 7], [9, 8], [9, 9]]
Evaluation Episode 9: Reward = 83, Steps = 18, Path: [[0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [6, 0], [7, 0], [8, 0], [9, 0], [9, 1], [9, 2], [9, 3], [9, 4], [9, 5], [9, 6], [9, 7], [9, 8], [9, 9]]
Evaluation Episode 10: Reward = 83, Steps = 18, Path: [[0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [6, 0], [7, 0], [8, 0], [9, 0], [9, 1], [9, 2], [9, 3], [9, 4], [9, 5], [9, 6], [9, 7], [9, 8], [9, 9]]

Evaluation Results:
Average Reward: 83.00
Average Steps: 18.00
Success Rate: 100.00%
```

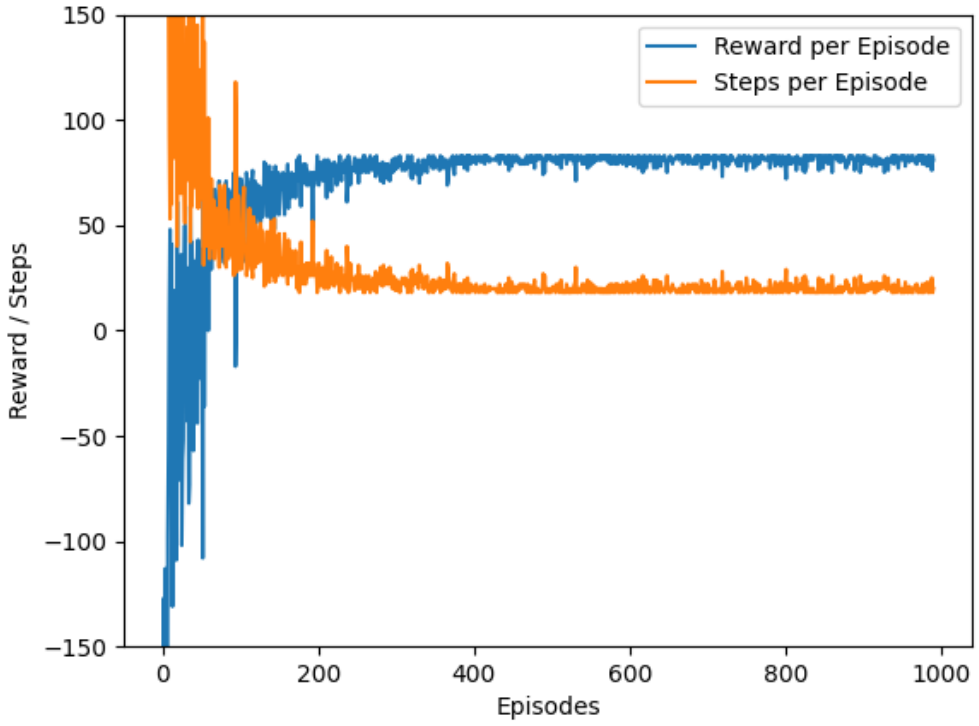
הצגה גרפית:

רף תגמול:(Reward per Episode)

- ניתן לראות את ההתייצבות של התגמול על הערכים האופטימליים לאחר הלמידה.
- בתחילת האימון התגמול היה נמוך יותר, אך לאורך הזמן הוא עלה בהדרגה והגיע לערך ממוצע של 83.

גרף צעדים:(Steps per Episode)

- הגרף מראה כיצד מספר הצעדים הצטמצם לאורך האימון.
- בסוף האימון, הסוכן הגיע ליעד תוך 18צעדים בממוצע, מה שמצביע על למידת מסלול יעיל.



הרצה 2: אימון לא מוצלח, עם תיקון המסלול בהערכה.

פרמטרים שהוזנו למערכת

```
# הגדרות
env = GridWorld()
state_dim = 2
action_dim = 4
lr = 0.001
gamma = 0.8
epsilon = 1.0
epsilon_decay = 0.995
epsilon_min = 0.4
batch_size = 32
memory = []
max_memory = 1000
```

1. **Epsilon (ε):**
   - ערך התחלתי ε = 0.8 ,

     הסוכן בוחר כמעט תמיד פעולה אקראית.
   - מה זה אומר:
     - במהלך האימון,

       הסוכן לא מתמקד בלמידה מהפעולות הקודמות,

       אלא בוחן הרבה אופציות בצורה אקראית.

       זה עשוי להוביל לאימון לא אפקטיבי,

       כשהמודל לא מצליח למצוא את המסלול האופטימלי.
     - חקירה רבה מדי יכולה למנוע מהסוכן ללמוד את המדיניות האופטימלית , כלומר, הוא יטעה שוב ושוב ולא ילמד את הדרך היעילה.

```
# אימון
for episode in range(1000):
```

2. **מספר האפיזודות:(Episodes)**
   - כמות 1000 אפיזודות
   - מה זה אומר:
     - מספר האפיזודות גבוה, אך בגלל שהסוכן בוחר פעולות אקראיות רוב הזמן, הוא לא מספיק ללמוד ולהתמקד במסלול האופטימלי.
     - ככל שהסוכן בוחר יותר פעולות אקראיות, הוא לא מצליח להפיק תועלת מהניסיון שנצבר במהלך האימון.

3. **תגמול:(Reward)**
   - התגמול הסופי לא עמד בציפיות, והתוצאות מצביעות על שיפור לא משמעותי במהלך האימון.

```
Episode 940, Reward: 70, Epsilon: 0.4000, Steps: 31
Best path: [[0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [4, 0], [4, 0], [5, 0], [5, 1], [5, 0], [5, 1], [5, 2], [5, 3], [4, 3], [5, 3], [6, 3], [7, 3], [7, 4], [7, 3], [7, 4], [8, 4], [9, 4], [9, 5], [9, 6], [9, 7], [9, 6], [9, 7], [9, 6], [9, 7], [9, 8], [9, 9]]
Episode 950, Reward: 71, Epsilon: 0.4000, Steps: 30
Best path: [[0, 0], [0, 1], [1, 1], [1, 2], [0, 2], [0, 3], [0, 4], [0, 5], [1, 5], [1, 6], [1, 7], [1, 8], [2, 8], [1, 8], [2, 8], [1, 8], [2, 8], [3, 8], [2, 8], [3, 8], [4, 8], [5, 8], [5, 9], [6, 9], [6, 8], [6, 9], [7, 9], [7, 8], [8, 8], [8, 9], [9, 9]]
Episode 960, Reward: 78, Epsilon: 0.4000, Steps: 23
Best path: [[0, 0], [0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [4, 1], [5, 1], [6, 1], [7, 1], [7, 0], [7, 1], [7, 2], [7, 3], [7, 4], [8, 4], [7, 4], [7, 5], [7, 6], [7, 7], [8, 7], [8, 8], [8, 9], [9, 9]]
Episode 970, Reward: 75, Epsilon: 0.4000, Steps: 26
Best path: [[0, 0], [0, 1], [0, 1], [0, 2], [0, 3], [0, 2], [0, 2], [1, 2], [2, 2], [3, 2], [4, 2], [5, 2], [5, 1], [5, 2], [6, 2], [6, 3], [7, 3], [7, 4], [8, 4], [8, 5], [8, 6], [8, 7], [8, 6], [8, 7], [8, 8], [8, 9], [9, 9]]
Episode 980, Reward: 44, Epsilon: 0.4000, Steps: 57
Best path: [[0, 0], [1, 0], [1, 1], [2, 1], [3, 1], [4, 1], [4, 0], [5, 0], [5, 1], [5, 0], [5, 1], [4, 1], [5, 1], [5, 2], [4, 2], [4, 1], [5, 1], [6, 1], [6, 2], [6, 1], [6, 2], [7, 2], [6, 2], [6, 1], [5, 1], [5, 0], [6, 0], [6, 0], [6, 1], [6, 2], [6, 3], [6, 4], [5, 4], [5, 5], [5, 6], [4, 6], [4,
Episode 990, Reward: 65, Epsilon: 0.4000, Steps: 36
Best path: [[0, 0], [0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [5, 0], [6, 0], [6, 1], [6, 2], [6, 1], [6, 2], [6, 1], [6, 2], [6, 3], [6, 4], [6, 5], [5, 5], [5, 6], [5, 5], [5, 6], [5, 5], [6, 5], [6, 4], [6, 5], [6, 6], [6, 7], [7, 7], [7, 6], [8, 6], [8, 7], [8, 6], [8, 7], [8, 8], [8, 9], [9,
Training complete!
```

4. **מספר הצעדים:(Steps)**
   - הסוכן לא הצליח לבצע את המסלול באופן אופטימלי. היו חזרות מיותרות במהלך הצעדים.

בס"ד

תוצאות האימון

- המסלול הנבחר :המסלול לא היה אופטימלי וכולל חזרות מיותרות על צעדים. לדוגמה, המסלול של אפיזודה 980 היה ארוך מאוד, עם יותר מ-50 צעדים, ולא הצליח להתייעל.

תוצאות ההערכה

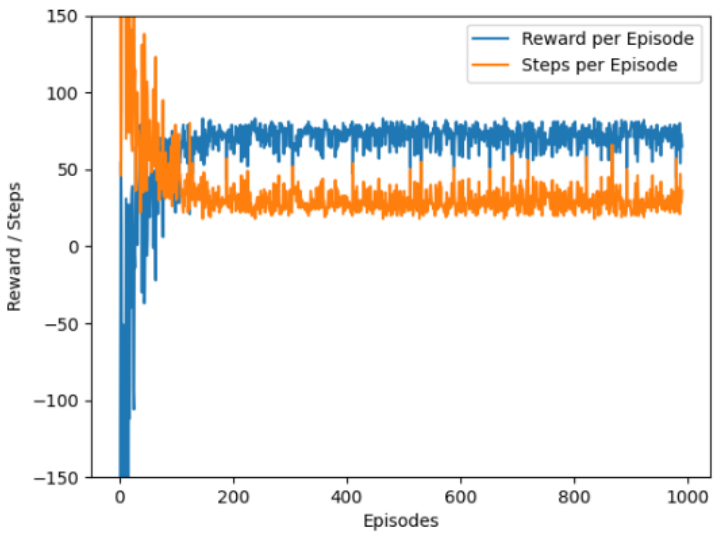
- לאחר סיום האימון, בוצעו 10 אפיזודות הערכה והתקבלו התוצאות הבאות:

```
Evaluation Episode 1: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [0, 7], [0, 8], [0, 9], [1, 9], [2, 9], [3, 9], [4, 9], [5, 9], [6, 9], [7, 9], [8, 9], [9, 9]]
Evaluation Episode 2: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [0, 7], [0, 8], [0, 9], [1, 9], [2, 9], [3, 9], [4, 9], [5, 9], [6, 9], [7, 9], [8, 9], [9, 9]]
Evaluation Episode 3: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [0, 7], [0, 8], [0, 9], [1, 9], [2, 9], [3, 9], [4, 9], [5, 9], [6, 9], [7, 9], [8, 9], [9, 9]]
Evaluation Episode 4: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [0, 7], [0, 8], [0, 9], [1, 9], [2, 9], [3, 9], [4, 9], [5, 9], [6, 9], [7, 9], [8, 9], [9, 9]]
Evaluation Episode 5: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [0, 7], [0, 8], [0, 9], [1, 9], [2, 9], [3, 9], [4, 9], [5, 9], [6, 9], [7, 9], [8, 9], [9, 9]]
Evaluation Episode 6: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [0, 7], [0, 8], [0, 9], [1, 9], [2, 9], [3, 9], [4, 9], [5, 9], [6, 9], [7, 9], [8, 9], [9, 9]]
Evaluation Episode 7: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [0, 7], [0, 8], [0, 9], [1, 9], [2, 9], [3, 9], [4, 9], [5, 9], [6, 9], [7, 9], [8, 9], [9, 9]]
Evaluation Episode 8: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [0, 7], [0, 8], [0, 9], [1, 9], [2, 9], [3, 9], [4, 9], [5, 9], [6, 9], [7, 9], [8, 9], [9, 9]]
Evaluation Episode 9: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [0, 7], [0, 8], [0, 9], [1, 9], [2, 9], [3, 9], [4, 9], [5, 9], [6, 9], [7, 9], [8, 9], [9, 9]]
Evaluation Episode 10: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [0, 7], [0, 8], [0, 9], [1, 9], [2, 9], [3, 9], [4, 9], [5, 9], [6, 9], [7, 9], [8, 9], [9, 9]]

Evaluation Results:
Average Reward: 83.00
Average Steps: 18.00
Success Rate: 100.00%
```

- במהלך האימון, הסוכן מבצע חקירה רבה של אפשרויות אקראיות, כלומר הוא בוחר פעולות בצורה חופשית ומדי פעם בוחר פעולה שמובילה לתוצאה פחות טובה. החקירה המופרזת גורמת לכך שהסוכן לא מצליח להתמקד בפתרון האופטימלי, ולמרות שהוא מבצע מספר גדול של אפיזודות, הוא לא מצליח ללמוד את הדרך היעילה ביותר.
- בשלב ההערכה, כשה Epsilon-קטן (כלומר, החקירה האקראית פוחתת), הסוכן כבר למד מהתנסויותיו את הערכים של הפעולות השונות והוא בוחר בצורה יותר חכמה את הפעולות שמובילות לתוצאות טובות יותר. כלומר, כל החלטה שהוא מקבל היא מבוססת על הידע שהוא רכש במהלך האימון.

הצגה גרפית:

- ניתן לראות בגרף את חוסר היציבות של הערכים , שמצביע על כך שהסוכן לא מצליח לבחור את המסלול האופטימלי. הסיבה לכך היא שהוא מבצע יותר מדי חיפושים אקראיים ולא מתמקד במה שלמד במהלך האימון. כתוצאה מכך, הסוכן לא מצליח לשפר את ביצועיו בצורה עקבית, ולא מצליח להתמקד במסלול היעיל ביותר.



הרצה 3: לא לתת לרובוט מספיק זמן ללמוד את המסלול, מספר נמוך מידי של חזרות לימוד.

```
# הגדרות
env = GridWorld()
state_dim = 2
action_dim = 4
lr = 0.001
gamma = 0.8
epsilon = 0.5
epsilon_decay = 0.995
epsilon_min = 0.4
batch_size = 32
memory = []
max_memory = 1000

# אימון
for episode in range(100):
```

פרמטרים שהוזנו למערכת:

1. **Epsilon (ε):**
   - הערך התחלתי היה 0.5, אך לא ירד מספיק לאורך האימון.
   - הסוכן בחר פעולות אקראיות ופעולות שמבוססות על מה שלמד,

     אך בגלל שהEpsilon לא ירד באופן מספק,

     רוב הזמן הסוכן לא התמקד בלמידה אפקטיבית.
2. **מספר האפיזודות:(Episodes)**
   - האימון כלל רק 100 אפיזודות.
   - מספר זה נמוך מדי

     כדי לא לאפשר לסוכן ללמוד את המסלול האופטימלי בצורה מלאה,

     והתוצאה היא שהוא לא הצליח להפיק את מלוא היתרון מהניסיון הנצבר.
3. **תגמול:(Reward)**
   - התגמול הסופי לא עמד בציפיות, והתוצאות מצביעות על שיפור לא משמעותי במהלך האימון
4. **מספר הצעדים:(Steps)**
   - במהלך האימון, הסוכן ביצע יותר צעדים ממה שצריך ולקח מסלול לא אופטימלי. זה נובע מהמגבלות של הזמן שניתן לו ללמוד ולשפר את הבחירות לאורך הזמן.

תוצאות האימון:

- במהלך האימון, הסוכן לא קיבל מספיק זמן ללמוד את הסביבה ולעבד את המידע בצורה שמביאה לתוצאות טובות. למרות שהוא ביצע מספר גבוה של אפיזודות, הוא לא בחר פעולות בצורה אופטימלית מכיוון ש Epsilon-לא ירד מספיק והוקצב לו זמן לימוד מצומצם.

```
Episode 40, Reward: -53, Epsilon: 0.8142, Steps: 154
Best path: [[0, 0], [0, 0], [0, 1], [0, 2], [0, 3], [0, 3], [0, 3], [0, 2], [0, 3], [1, 3], [0, 3], [0, 3], [0, 3], [0, 3], [1, 3], [1, 4], [2, 4], [2, 5], [2, 4], [2, 3], [3, 3], [3, 4], [2, 4], [3, 4], [4, 4], [3, 4], [4, 4], [4, 3], [3, 3], [4, 3], [4, 2], [5, 2], [6, 2], [7, 2], [7, 1], [6, 1], [7
Episode 50, Reward: 18, Epsilon: 0.7744, Steps: 83
Best path: [[0, 0], [1, 0], [1, 0], [1, 1], [2, 1], [1, 1], [0, 1], [0, 2], [0, 1], [0, 2], [1, 2], [2, 2], [3, 2], [4, 2], [4, 3], [5, 3], [4, 3], [4, 4], [3, 4], [3, 3], [4, 3], [5, 3], [4, 3], [5, 3], [6, 3], [5, 3], [5, 4], [6, 4], [6, 3], [5, 3], [6, 3], [6, 2], [5, 2], [5, 1], [5, 2], [5, 3], [5
Episode 60, Reward: 27, Epsilon: 0.7366, Steps: 74
Best path: [[0, 0], [0, 0], [1, 0], [2, 0], [2, 0], [3, 0], [3, 0], [3, 0], [2, 0], [2, 1], [2, 2], [1, 2], [1, 3], [1, 2], [0, 2], [0, 2], [0, 3], [0, 3], [1, 3], [1, 4], [0, 4], [0, 4], [0, 3], [1, 3], [2, 3], [2, 4], [2, 5], [2, 6], [2, 5], [1, 5], [1, 4], [2, 4], [2, 5], [1, 5], [0, 5], [0, 6], [0
Episode 70, Reward: 33, Epsilon: 0.7005, Steps: 68
Best path: [[0, 0], [0, 0], [1, 0], [2, 0], [2, 0], [3, 0], [3, 1], [2, 1], [2, 0], [1, 0], [1, 1], [1, 2], [0, 2], [0, 3], [1, 3], [1, 4], [2, 4], [2, 5], [3, 5], [3, 6], [4, 6], [4, 7], [5, 7], [6, 7], [5, 7], [6, 7], [7, 7], [7, 6], [7, 5], [6, 5], [6, 6], [7, 6], [7, 7], [7, 6], [7, 5], [7, 6], [8
Episode 80, Reward: 72, Epsilon: 0.6663, Steps: 29
Best path: [[0, 0], [0, 0], [0, 1], [0, 0], [0, 1], [1, 1], [2, 1], [3, 1], [3, 2], [2, 2], [2, 1], [3, 1], [3, 2], [4, 2], [5, 2], [5, 1], [6, 1], [6, 2], [6, 3], [5, 3], [5, 4], [6, 4], [6, 5], [6, 6], [6, 7], [6, 8], [7, 8], [7, 9], [8, 9], [9, 9]]
Episode 90, Reward: 62, Epsilon: 0.6337, Steps: 39
Best path: [[0, 0], [0, 1], [0, 2], [0, 3], [1, 3], [1, 2], [1, 1], [0, 1], [0, 1], [0, 2], [0, 1], [0, 2], [1, 2], [2, 2], [2, 1], [3, 1], [2, 1], [3, 1], [4, 1], [4, 2], [4, 3], [5, 3], [6, 3], [5, 3], [5, 4], [6, 4], [6, 5], [6, 4], [6, 5], [6, 6], [7, 6], [7, 5], [7, 6], [7, 7], [6, 7], [7, 7], [7
Training complete!
```

בס"ד

תוצאות ההערכה:

- בהערכה, ניתן לראות כי המסלול שבחר הסוכן היה אופטימלי. הסוכן לא היה נתון למבצעי חקר (Exploration) במידה רבה כמו במהלך הלמידה, והיה מסוגל לנצל את המידע שלמד במהלך האימון כדי לבחור את הצעד האופטימלי

```
Evaluating Performance...

Evaluation Episode 1: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [1, 6], [2, 6], [3, 6], [4, 6], [4, 7], [5, 7], [6, 7], [7, 7], [8, 7], [8, 8], [9, 8], [9, 9]]
Evaluation Episode 2: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [1, 6], [2, 6], [3, 6], [4, 6], [4, 7], [5, 7], [6, 7], [7, 7], [8, 7], [8, 8], [9, 8], [9, 9]]
Evaluation Episode 3: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [1, 6], [2, 6], [3, 6], [4, 6], [4, 7], [5, 7], [6, 7], [7, 7], [8, 7], [8, 8], [9, 8], [9, 9]]
Evaluation Episode 4: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [1, 6], [2, 6], [3, 6], [4, 6], [4, 7], [5, 7], [6, 7], [7, 7], [8, 7], [8, 8], [9, 8], [9, 9]]
Evaluation Episode 5: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [1, 6], [2, 6], [3, 6], [4, 6], [4, 7], [5, 7], [6, 7], [7, 7], [8, 7], [8, 8], [9, 8], [9, 9]]
Evaluation Episode 6: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [1, 6], [2, 6], [3, 6], [4, 6], [4, 7], [5, 7], [6, 7], [7, 7], [8, 7], [8, 8], [9, 8], [9, 9]]
Evaluation Episode 7: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [1, 6], [2, 6], [3, 6], [4, 6], [4, 7], [5, 7], [6, 7], [7, 7], [8, 7], [8, 8], [9, 8], [9, 9]]
Evaluation Episode 8: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [1, 6], [2, 6], [3, 6], [4, 6], [4, 7], [5, 7], [6, 7], [7, 7], [8, 7], [8, 8], [9, 8], [9, 9]]
Evaluation Episode 9: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [1, 6], [2, 6], [3, 6], [4, 6], [4, 7], [5, 7], [6, 7], [7, 7], [8, 7], [8, 8], [9, 8], [9, 9]]
Evaluation Episode 10: Reward = 83, Steps = 18, Path: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [1, 6], [2, 6], [3, 6], [4, 6], [4, 7], [5, 7], [6, 7], [7, 7], [8, 7], [8, 8], [9, 8], [9, 9]]

Evaluation Results:
Average Reward: 83.00
Average Steps: 18.00
Success Rate: 100.00%
```

הסבר על הבדל בין למידה להערכה:

במהלך הלמידה:

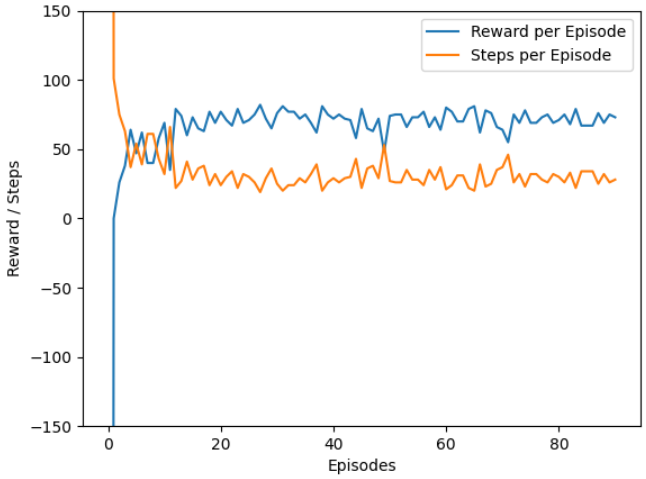
- הסוכן לא הצליח ללמוד טוב את הסביבה. הוא לא שיפר את ההחלטות שלו מספיק כי ה-Epsilonלא ירד מספיק לאורך הזמן.
- כשה Epsilon-היה גבוה, הסוכן עשה הרבה פעולות אקראיות כדי לחקור את הסביבה, במקום להתמקד בלמידה מהפעולות שעשה קודם. זה גרם לו לבחור מסלול פחות טוב.

במהלך ההערכה:

- אחרי שלמד מספיק על הסביבה, הסוכן לא עשה יותר מדי פעולות אקראיות, והשתמש במידע שלמד כדי לקבל החלטות טובות יותר.
- כש Epsilon היה נמוך יותר, הסוכן עשה פחות פעולות אקראיות והצליח לבחור את הפעולות הטובות ביותר, לפי מה שלמד.
- בסופו של דבר, הסוכן בחר את המסלול הטוב ביותר בהערכה, אבל לא הצליח לעשות את זה בלמידה, כי לא היה לו מספיק זמן ללמוד.

הצגה גרפית:

- ניתן לראות בבירור שלא היה לרובוט מספיק חזרות לימוד על מנת ללמוד את הסביבה .



טבלת השוואה בין האלגוריתמים.

Q-Learning	DQN (Deep Q-Network)	פרמטר
(0, 0)	(0, 0)	**התחלה**
(9, 9)	(9, 9)	**מיקום יעד**
חיובי ביעד, שלילי במכשולים	חיובי ביעד, שלילי במכשולים	**ערך תגמול**
מוגדר מראש	משתנה במהלך האימון	**(Alpha) שיעור למידה**
0.9	0.9	**(Gamma) מקדם הנחה**
קבוע	קטן בהדרגה	**Epsilon (Exploration)**
עדכון ישיר של ערכי הטבלה	שימוש ברשת נוירונים לניבוי ערכים	**שיטת עדכון**
פחות זמן ריצה ,תלוי בגודל הבעיה	יותר זמן ריצה, תלוי בגודל הרשת	**זמן ריצה**
מוגבל לסביבות פשוטות ידועות מראש	יכול להתמודד עם סביבות מורכבות שיכולות להשתנות בזמן הריצה	**יכולת התמודדות עם סביבות מורכבות**
דרישות מינימליות	דורש הרבה משאבים	**דרישות מערכת**
אין אופטימיזציה מתקדמת	משתמש באופטימיזר לשיפור האימון	**אופטימיזציה**
קבוע	מתדרדר מ-1 ל-0.1	**(Epsilon) הסתברות אקראית**
מוגדרות כערכים 100-	נלמדים מתוך הממשק עם הסביבה	**מכשולים**
תגמול מקסימלי ביעד, שלילי בשאר הפעולות	תגמול מנוהל רנדומלית עם הרשת	**פונקציה לתגובה/תגמול**
רק טבלה.	רשת נוירונים חבויה עם שכבות מרובות	**רשת הנויטרונית**
דורש פחות משאבים, יכול להיות איטי	דורש יותר משאבים, יעיל יותר בסביבות מורכבות	**אימון ודיוק**
מוגבל לסביבות פשוטות	מתמודד טוב יותר עם סביבות מורכבות	**יכולת התמודדות עם סביבות מורכבות**