תרגיל 2 – תכנון אוטומטי ולמידת חיזוקים

גרסת ג'יפיטי::  
  
  
  
  
**הוראות עבודה לתרגיל 2 – תכנון אוטומטי ולמידת חיזוקים**

**חלק 1: הכנה לסביבת העבודה**

1. **קריאת מידע על Open AI Gym**
   * קראו על סביבת Gymnasium (גרסה עדכנית של Open AI Gym) באתר: [https://gymnasium.farama.org](https://gymnasium.farama.org/).
   * השתמשו במידע מהאתר או ממקורות נוספים באינטרנט לפי הצורך.
2. **התקנת Gymnasium**
   * התקינו את הסביבה במחשבכם על מנת שתוכלו להשתמש בה.
3. **שפת התכנות**
   * בצעו את המשימה בשפת Python.

**חלק 2: פתרון בעיית *Frozen Lake***

1. **בחירת הבעיה**
   * בחרו את בעיית **Frozen Lake** מתוך תת-סביבה toy-text.
   * עבדו עם לוח בגודל 4x4 כאשר הפרמטר is\_slippery מוגדר כ-True.
2. **שימוש באלגוריתם R-Max**
   * פתרו את הבעיה באמצעות האלגוריתם **R-Max**.
   * השתמשו בערך הנחה γ=0.95\gamma = 0.95.
3. **מימוש שיטות פתרון**  
   א. **Modified Policy Iteration (MPI)**
   * בחרו תנאי עצירה לפי שיקול דעתכם.  
     ב. **Real-Time Dynamic Programming (RTDP)**
   * השתמשו במדיניות בחירה *ε-greedy* בכל איטרציה.

**חלק 3: הערכת מדיניות**

* **מימוש First-Visit Monte Carlo (FVMC)**
  + חשבו את ערך המדיניות בעולם האמיתי (לא במודל הפיקטיבי של R-Max).

**חלק 4: דרישות הקוד**

1. **מימוש פונקציות מרכזיות**
   * עליכם לממש את השיטות:
     + r\_max, mpi, rtdp (כל אחת מהן מחזירה מדיניות).
   * פונקציות נוספות:
     + **draw\_policy**: מציירת את המדיניות כטבלת 4x4, כאשר בכל תא מצוינת הפעולה שיש לבצע במצב זה.
     + **draw\_value**: מציגה פונקציית ערך כטבלת 4x4, כאשר בכל תא מוצג הערך של המצב.
     + **fvmc**: מחשבת את ערך כל מצב באמצעות *First-Visit Monte Carlo*.
2. **יעילות קוד**
   * יש לתת דגש על יעילות האלגוריתם, למשל באמצעות אתחול חכם של חישוב פונקציית הערך.
3. **מדידת ביצועים**
   * הגדירו שיטה בשם **performance(x)** שתבצע את הפעולות הבאות:  
     א. תריץ את אלגוריתם R-Max עם MPI.  
     ב. כל kk צעדי סימולציה, תפעילו את fvmc על מנת לחשב את ערך המדיניות הנוכחית במצב (0,0).  
     ג. צרו גרף:
     + ציר XX: מספר צעדי הסימולציה.
     + ציר YY: ערך המצב (0,0).
     + כל kk צעדים הוסיפו נקודה לגרף.  
       ד. דוגמה: עבור k=1000k=1000, חישבו ערך בכל 1000 צעדים, כך שיתקבל גרף עם לפחות 20 נקודות עד להתכנסות לערך יציב.

**חלק 5: הגשה**

1. **מבנה הקובץ**
   * הגישו קובץ ZIP בשם assignment1\_<id1>\_<id2>.zip.
   * הקובץ יכלול:  
     א. **קוד מתועד**: עם כל השיטות שפורטו לעיל.  
     ב. **קובץ PDF**:
     + פרטים אישיים (ת.ז ושמות).
     + הסבר כללי על הקוד והשיטות המרכזיות.
     + תיאור שיפורים אם בוצעו.
     + גרף מתוצאות performance(x).
     + הדפסת המדיניות והערך שלה בשלב ההתכנסות.
2. **כלים להצגת גרפים**
   * השתמשו בספריית **Matplotlib** ליצירת גרפים.

**חלק 6: קריטריונים להערכה**

* נכונות הקוד.
* יעילות האלגוריתמים.

בהצלחה!

גרסה מקורית:

* לקרוא על סביבת  [Open AI Gym](https://gym.openai.com/) ולהתקין אותה בכדי שתוכלו להשתמש בה. הסביבה נקראת היום: Gymnasium ונמצאת ב- <https://gymnasium.farama.org> יש שם את כל המידע שאתם צריכים כדי להשתמש בסביבה, אבל תוכלו למצא מידע נוסף באתרים שונים, אם תרצו.
* שפת התכנות: פייטון.
* יש לפתור את בעיית frozen-lake מתוך סביבת toy text. פתרו את בעיית ה4\*4 כאשר is\_slippery מקבל ערך true ע״י שימוש באלגוריתם r\_max לפתרון הבעיה. השתמשו בגאמה=0.95. כדי לפתור מודל נתון ממשו שתי שיטות פתרון:

א. modified policy iteration. ביחרו תנאי עצירה כרצונכם.ן.

ב. real-time dynamic programming. המדיניות שלפיה תבצעו את בחירת המסלול בכל איטרציה תיהיה אפסילון-חמדנית.

* ממשו שיטת first-visit monte carlo על מנת להעריך את הערך של מדיניות בעולם האמיתי (כלומר, לא במודל הפיקטיבי של r\_max).

עליכם להגיש קוד מתועד, וצריך להכיל גם את השיטות הבאות: r\_max, mpi, rtdp. שלוש השיטות הנ״ל צריכות להחזיר מדיניות. כתבו גם שיטות draw-policy, draw-value ו-fvmc. הראשונה מקבלת מדיניות ומציירת אותה כטבלה בגודל הלוח – כלומר טבלה בגודל 4x4 כך שבכל כניסה בטבלה כתוב כיוון התנועה של המדיניות כאשר הסוכן נמצא במיקום הנ״ל בלוח. השנייה מקבלת פונקציית ערך ומציגה אותה באותה צורה, רק שבמקום הפעולה יופיע הערך של כל מצב. האחרונה מחשבת את ערך כל מצב בעזרת first-visit MC. עיקר הניקוד יהיה על נכונות הקוד, אבל גם יעילות האלגוריתם תילקח בחשבון (למשל, ייעול חישוב פונקצית הערך על ידי אתחול טוב).

כמו כן יש להגדיר שיטה בשם performance(x) שתייצר גרף של שיפור הביצועים של האלגוריתם תוך כדי למידה במסגרת r\_max עם mpi. שיטה זאת תריץ את האלגוריתם, וכל k צעדי סימולציה (לא צעדי חישוב) השיטה תפעיל את fvmc על מנת לחשב את ערך המדיניות הנוכחית במצב (0,0) ואז תייצר גרף שמציג את ערך המצב הזה כפונקציה של מספר צעדי הסימולציה. כלומר, ציר X יהיה מספר הצעדים, וציר Y יהיה הערך, והגרף יכיל נקודה עבור כל כפולה של k. למשל, אם k=1000 אז כל 1000 צעדי סימולציה נעצור ונחשב את הערך v של המדיניות הנוכחית במצב (0,0). אם לדוגמה אחרי 5000 צעדים הערך הוא 10, אז בגרף תופיע נקודה ב (5000,10). ניתן להשתמש בספריית matplotlib.

יש להגיש קובץ שיקרא assignment1\_<id1>\_<id2>.zip, המכיל את קבצי הקוד יחד עם מסמך pdf קצר בו ת.ז, השמות שלכם, הסבר על הקוד בכלליות, והסבר על שיטות מרכזיות. אם יעלתם את האלגוריתם בצורה כלשהי, הסבירו זאת בקובץ הנ״ל. קובץ ה-pdf יכיל גם גרף שייוצר על ידי performance(x). בחרו x סביר כך שיהיו בגרף שלכם לפחות 20 נקודות עד שרואים התכנסות לערך יציב פחות או יותר. כמו כן, הדפיסו את המדיניות ואת הערך שלה עבור המדיניות שמתקבלת בשלב זה.

בהצלחה!