תרגיל 6 | מבוא לבינה מלאכותית

שחר כהן ודוד רופין

שאלה 8 – מהי הסיבוכיות של אלגוריתם ה-value iteration משאלה 1?

להלן המימוש שלנו לאלגוריתם value-iteration:

```
def _train(self):
    """
Sets the values in the state-value dictionary
"""
    for _ in range(self.iterations):
        cur_iteration_values = util.Counter()
        for state in self.mdp.getStates():
            if self.mdp.isTerminal(state):
                # the expected return of terminal states is zero
                continue
            cur_val = float('-inf')
            for action in self.mdp.getPossibleActions(state):
                q_val = self.getQValue(state, action)
                if q_val > cur_val:
                    cur_val = q_val
            cur_iteration_values[state] = cur_val
        self.values = cur_iteration_values
```

בכל איטרציה, עבור כל מצב במשחק, אנחנו בודקים האם זה מצב סיום בזמן קבוע. אם זה לא מצב סיום, אנחנו לוקחים את ערך ה-q-value המקסימלי עבור הפעולות החוקיות במצב הנוכחי. ה-q-value בהינתן מצב ופעולה הוא התוחלת של ערכי המצבים הבאים שאנחנו יכולים להגיע אליהם, בהינתן שביצענו את הפעולה הנ"ל:

```
def getQValue(self, state, action):
    """
  The q-value of the state action pair
  (after the indicated number of value iteration
  passes).  Note that value iteration does not
  necessarily create this quantity and you may have
  to derive it on the fly.
"""
    "*** YOUR CODE HERE ***"
    q_value = 0
    for next_state, prob in self.mdp.getTransitionStatesAndProbs(state, action):
        reward = self.mdp.getReward(state, action, next_state)
        q_value += prob * (reward + self.discount * self.values[next_state])
    return q_value
```

נסמן את מספר הפעולות שניתן לבצע ממצב מסוים במשחק ב-A. מספר המצבים הבאים אליהם אנחנו יכולים להגיע ממצב מסוים בלוח זהה למספר הפעולות, ולכן גם הוא יסומן ב-A. סיבוכיות חישוב ה-q-value, אם כך, היא O(A) (במקרה הזה בתוחלת, בגלל שימוש בטבלת גיבוב).

נסמן את מספר האיטרציות ב-t ואת מספר המצבים במשחק ב-S. כיוון שבכל איטרציה עבור כל מצב במשחק אנחנו מחשבים את מקסימום ה-q-value של כל אחת מהפעולות החוקיות, נקבל כי סיבוכיות האלגוריתם בסה"כ היא $O(t \cdot S \cdot A^2)$ (במקרה הזה בתוחלת, בגלל שימוש בטבלת גיבוב).

שאלה 9

Discount Factor – הפרמטר הנ"ל משפיע על כמה הרווח העתידי חשוב לאלגוריתם. ככל שהDiscount Factor נמוך יותר, האלגוריתם יתחשב פחות ברווח העתידי בסוף המסלול. בבעיה שלנו, כאשר שמנו discount factor נמוך יותר ובהינתן שיש רעש מסוים במשחק, האלגוריתם העדיף להימנע מהדרך המסוכנת וללכת בדרך הבטוחה יותר. בנוסף, ב-Discount Factor נמוך מאוד, האלגוריתם יעדיף להשיג רווחים קרובים יותר גם אם הם נמוכים בהשוואה לרווחים רחוקים יותר (במקרה שלנו – העדיף ללכת למשבצת שמרוויחים בה נקודה במקום למשבצת שמרוויחים בה 10 נקודות).

Noise – ככל שהרעש היה גבוה יותר, האלגוריתם נמנע מהדרך המסוכנת והעדיף ללכת בדרך בטוחה יותר. אם ה-Discount Factor גבוה וה-Noise גבוה יחסית, האלגוריתם יתחשב גם בסיכון ארוך-הטווח ולא רק בסיכון קצר-הטווח. ככל שה-Noise נמוך יותר, קל יותר לשלוט באלגוריתם (הוא יתכנס מהר יותר לערכים האופטימליים).

Life Reward – ככל שה-life reward גבוה ביחס לreward המקסימלי שהשחקן מקבל בסיום המשחק, השחקן ישאף להתמקם במשבצות "יציבות" שמרחיקות אותו מסיום המשחק (כלומר שלא קרובות ל"צוק" ושהסיכוי להגיע מהן בצעדים בודדים למשבצת הסיום ע"י תנועה אקראית נמוך).

שאלה 10

כן. בשיעור דיברנו על exploration-exploitation tradeoff, כלומר על האיזון שבין חקירת המרחב לבין הגברת הדיוק של הידע הקיים. ככלל, בתחילת האימון נעדיף שהאלגוריתם יחקור את המרחב, וככל שהאימון מתקדם והידע שרכשנו על המרחב גדול יותר, נעדיף לדייק את הידע שלנו על המסלול האופטימלי. במקום להשתמש בשיטת epsilon-greedy, אפשר להשתמש בשיטת למידה דומה, בה אנחנו מקטינים את epsilon ע"י הכפלה בפקטור כלשהו לאחר כל איטרציה. כך, בשלבים מתקדמים יותר של האימון האלגוריתם יתמקד בדיוק, ובשלבים ההתחלתיים יתמקד בחקירה.