

Passive Reinforcement Learning

באלגוריתם הכי פשוט, אנחנו מנסים ללמוד את הפונקציה Π באמצעות ניסוי וטעיה - מנסים לבצע פעולות, רואים מה התוצאה ומה הreward, ולפי זה בוחרים את ה Π הכי טוב. אבל גם בpassive יש אלגוריתמים יותר חכמים. למשל, במקום לבנות את Π עצמה בתור ניסוי וטעיה, אפשר לחפש את התכונות של המודל עצמו:

- אפשר לנסות ללמוד את R - לרשום את התמורה שמקבלים בכל מצב. התמורה אינה הסתברותית.
 - אפשר לנסות ללמוד את T - פונקציה הtransition.
- אחרי שיודעים את R ואת T , אפשר להשתמש באלגוריתמי חיפוש (policy iteration או value iteration) בשביל לחשב את הפונקציה האופטימלית Π^* .
בסופו של דבר Passive Reinforcement Learning הוא די מוגבל - תמיד את אותו Π .

Active Reinforcement Learning

Active RL אין Π קבוע - מחליטים כל הזמן על הפעולה הבאה בצורה אקטיבית(מה כדאי לעשות הלאה), בהסתמך על הידע שצברנו במהלך הריצה. המטרה נשארה אותו דבר - למצוא את ה Π האופטימלי. תזכורת - אנחנו מחפשים תועלת אופטימלית, לפי משוואת בלמן:

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U(s')$$

אנחנו רוצים למקסם את התועלת, כלומר לבחור:

$$a = \operatorname{argmax}_a \left(\sum_{s'} T(s, a, s') U(s') \right)$$

אלגוריתם חמדני

האלגוריתם בוחר בכל שלב את הפעולה שהיא הכי טובה בהתחשב במה שהוא ניסה עד עכשיו.

- מתחילים בלשייך תמורה 0 לכל הפעולות
 - בכל פעם בוחרים את הפעולה עם התמורה הכי טובה
 - בכל פעם שעושים פעולה מעדכנים את התמורות
- זה בעייתי - כי עלולים להגיע בקלות למקסימום לוקלי.

Exploration & Exploitation

לסוכן לומד יש 2 מטרות:

- ניצול(Exploitation):

- מיקסום התמורה על כל פעולה
- לפעול לפי מה שכבר ידוע

• חקירה (Exploration):

– מיקסום התועלת לטווח הארוך

– לפעול כדי לנסות ללמוד מצבים שאנחנו עוד לא מכירים

Exploitation טהור מסתכן בלהתקע במקסימום לוקלי. Exploration טהור הוא חסר ערך, כי הוא לא משתמש בידע. לכן בד"כ נרצה לשלב.

איך משלבים?

- לבחור בהסתברות $\frac{1}{\epsilon}$ פעולה אקראית (בשביל exploitation) ובשאר ההסתברויות את greedy. הרעיון הוא שככל שעובר הזמן למדנו יותר ולכן פחות חשוב להמשיך ללמוד.
- לבחור לפי התועלת - פעולות יותר טובות יקבלו הסתברות יותר טובה.

אלגוריתם Q-Learning

זה האלגוריתם הכי נפוץ של Reinforcement Learning. הרעיון: לחשב ערך לכל מצב ופעולה $Q(s, a)$ - ערך שמדבר על הexpected utility כאשר אנחנו נמצאים במצב s ומבצעים פעולה a . לפי זה אפשר לחשב:

$$\Pi^*(s) = \operatorname{argmax}_{a \in A} Q(s, a)$$

- מתחילים עם ערך שרירותי קבוע - למשל $Q(s, a) = 0 \forall s, a$
- בכל episode (רצף של מצבים ופעולות ממצב התחלתי עד למצב סופי):

– בכל שלב של episode:

- * בוחרים a בהינתן s באמצעות מדיניות שנגזרת מ Q (תלוי אלגוריתם¹)
 - * מבצעים את הפעולה. לומדים מה התרומה r ומה המצב שהגענו אליו s' .
 - * מעדכנים: $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
 - למה לא $Q(s, a) \leftarrow R(s') + \gamma \max_{a'} Q(s', a)$? כי לא בהכרח תמיד מגיעים לאותו s' ! לכן משתמשים בקבוע $\alpha \leq 1$, שקובע את קצב הלמידה:
 - (אם $\alpha = 1$ אז $Q(s, a) \leftarrow R(s') + \gamma \max_{a'} Q(s', a) - Q(s, a)$ כלומר מתעלמים מהידע הקודם על אותו מצב.)
 - נשים \heartsuit : a' שלפיו מבצעים את העדכון (כלומר הפעולה הטובה ביותר שאפשר לבצע בשלב הבא) היא לא בהכרח הפעולה שבאמת נבצע בשלב הבא!
 - מסתכלים רק שלב אחד קדימה, כי לא יודעים לאן נגיע בשלב הבא. האלגוריתם לא לומד את T בצורה מפורשת. T וצפיית השלבים הבאים משוקללים לתוך $Q(s, a)$.
- חוזרים על זה שוב ושוב עד שמגיעים למצב סופי.

תכונות של Q-Learning

- לא לומד את המודל (אולי למדנו את R , אבל לא את T)

- מתכנס לאט לאט - אבל מתכנס

¹הכי פשוט לבחור את $\operatorname{argmax}_{a \in A} Q(s, a)$ אבל זה לא חייב בהכרח להיות הפעולה האופטימלית (לפי Q)! בדרך כלל נרצה לעשות גם

אלגוריתם SARSA

הרעיון: 1. נמצאים במצב s

2. בוחרים פעולה a

3. עוברים למצב s'

4. בוחרים פעולה a'

5. מעדכנים את $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma Q(s', a') - Q(s, a)]$

כלומר, לא יכול להיות שעדכנו את $Q(s, a)$ עם a' אחד ובשלב הבא בחרנו פעולה אחרת.