



קריית החינוך  
פארק המדע  
בית לערכים  
למצוינות ולחדשנות

# Model Free TD - Temporal Difference

SARSA  
Q-Learning

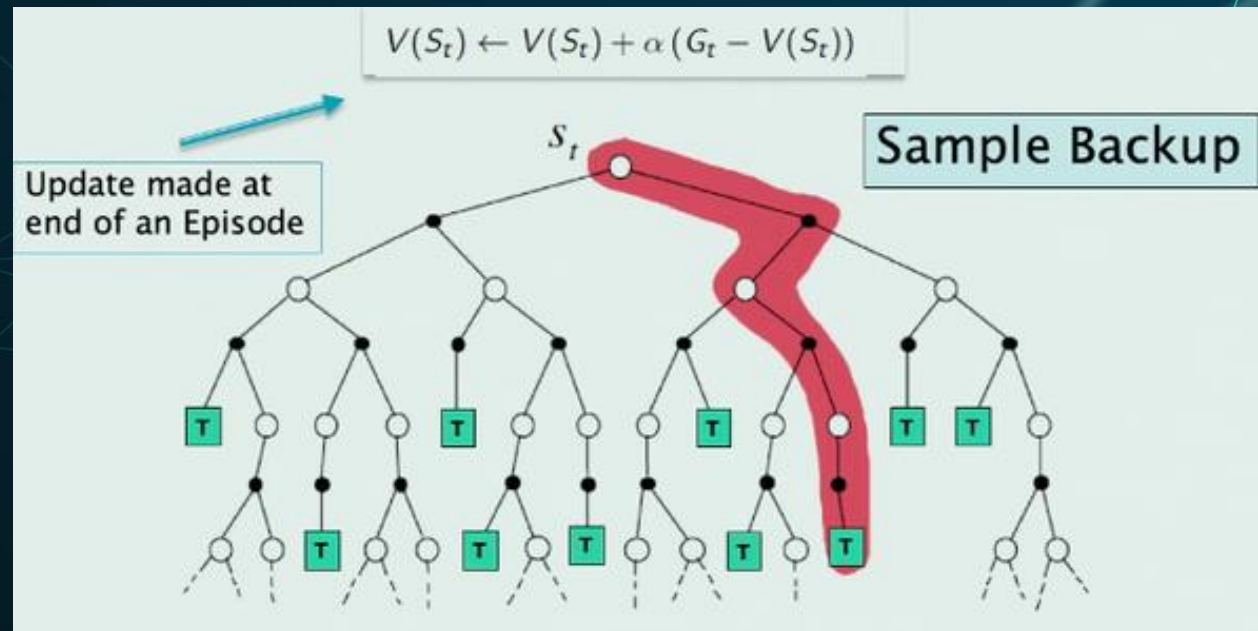


# תזכורת - Monte Carlo Method

- במונטה קרלו אנחנו מבצעים דגימה של הסביבה עד למצב סופי (T) ושומרים את המסלול. בסיום המסלול אנחנו יכולים לחשב:

$$V(s_t) = V(s_t) + \alpha [G_t - V(s_t)]$$

$$G_t = R_t + \gamma R_{t1} + \gamma^2 R_{t2} + \dots + \gamma^T R_T$$



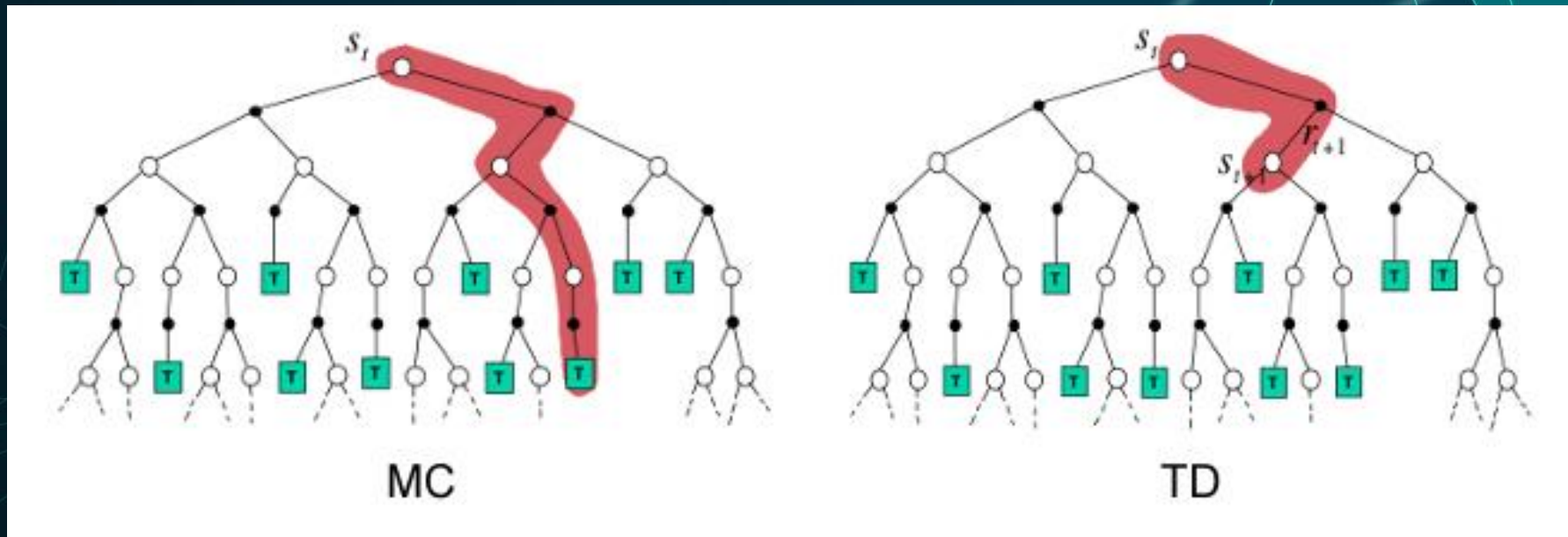


# החסרונות של MC

- בסביבה שבה המהלכים עד למצב סופי הם ארוכים מאוד – מונטה קרלו לא יעיל.
- למשל במשחק שחמט יש מהלכים רבים מאוד עד שמקבלים את המצב הסופי. הלימוד עם מונטה קרלו כמעט בלתי אפשרי.
- בסביבה שבה המהלכים אין סופיים לא ניתן כלל להשתמש במונטה קרלו.
- סוכן המשקיע בבורסה אין מצב סופי ולכן לא ניתן להשתמש באלגוריתם.
- גם במשחקים בהם יש מצב סופי אך ניתן להיכנס ללולאות אין סופיות – נתקשה להשתמש באלגוריתם.

# הפתרון Temporal Difference

- ב TD אנחנו דוגמים את הסביבה כמו ב MC אבל במקום לחכות לסוף המהלך (למצב סופי) אנחנו מעדכנים את הערכים מייד בסיום כל מצב.
- הדרך לעשות זאת היא באמצעות שימוש בנוסחת בלמן.





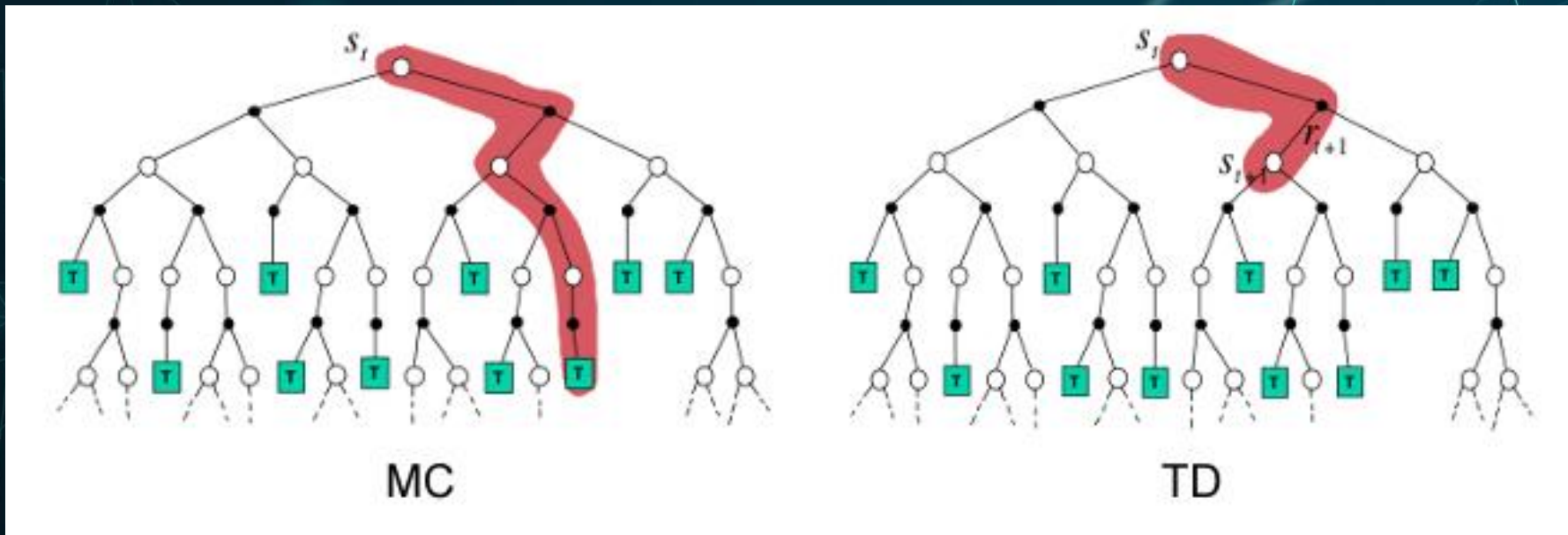
# הפתרון Temporal Difference

$$V(s_t) = V(s_t) + \alpha [G_t - V(s_t)]$$

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots + \gamma^T R_T$$

$$G_t = R_t + \gamma V(s_{t+1})$$

$$V(s_t) = V(s_t) + \alpha [R_t + \gamma V(s_{t+1}) - V(s_t)]$$



# TD- Bootstrapping

$$V(s) = V(s) + \alpha[R + \gamma V(s') - V(s)]$$

- TD עושה שימוש ב Bootstrapping שכן הוא מעדכן את ערך המצב בהסתמך על ערך המצב הבא, שהוא נקבע על ידיו.
- החישוב בסוגריים נקרא בספרות TD error:

$$TD\ error = R + \gamma V(s') - V(s)$$



# Model Free Q-Table

- בדומה ל MC כדי להתגבר על הבעייתיות של מודל לא ידוע אנחנו נעשה שימוש בטבלת Q (מצב-פעולה) לחישוב הערכים.
- בדרך זו אנחנו נוכל להפיק את המדיניות מהטבלה.
- בהינתן מצב מסויים אנחנו יכולים באמצעות טבלת Q לאתר את הפעולה שתביא לערך הגבוה ביותר, גם אם אין בידינו את המודל.

$$P(s) = \operatorname{argMax}_a(Q(s, a))$$

$$V(s) = \max_a(Q(s, a))$$



# MC vs. TD

• ב MC עדכון טבלת הערכים נעשית לפי הנוסחה הבאה:

$$V(s) = V(s) + \alpha(G_t - V(s))$$

$$G_t = R_0 + \gamma R_1 + \gamma^2 R_2 + \dots + \gamma^T R_T$$

$$Q(s, a) = Q(s, a) + \alpha(G_t - Q(s, a))$$

• ב TD נבצע עדכון של הטבלה באמצעות Bootstrap:

$$V(s) = V(s) + \alpha(R + V(s') - V(s))$$

$$Q(s, a) = Q(s, a) + \alpha(R + \gamma Q(s', a') - Q(s, a))$$

• Sutton הוכיח בשנת 1988 כי גם האלגוריתם של TD מתכנס ונותן לנו את  $Q^*$ .



## ביצוע הדגימה של הסביבה

$$Q(s, a) = Q(s, a) + \alpha(R + \gamma Q(s', a') - Q(s, a))$$

- בדומה ל MC אנחנו צריכים לדגום את הסביבה כדי לקבל את הצעד הבא  $R, S'$ .
- בעת הדגימה עלינו להתחשב בדילמה **Exploration vs Exploitation**
- בספרות קיימים שני אלגוריתמים כמעט זהים לביצוע הדגימה ממשפחת TD:
  - אלגוריתם SARSA
  - אלגוריתם Q-learning.

אלגוריתמים אילו מכונים TD(0) – כיוון שאנחנו דוגמים בכל פעם צעד אחד קדימה (לעומת MC בו דוגמים את כל המהלך עד לסוף המשחק).

# אלגוריתם SARSA

- הסוכן נמצא במצב  $S$  יבחר את הפעולה  $A$  בהתאם לטבלת  $Q$  ולמדיניות  $\epsilon$ -greedy.
- הסוכן ידגום את הסביבה ויקבל את התוצאות הבאות:  $S, A, R, S', A'$ .
- בחירת  $A'$  תעשה לפי המדיניות הנוכחית שלו  $Q$ -Table &  $\epsilon$ -greedy.
- אחרי כל צעד ישתמש הסוכן בנתונים כדי לעדכן את טבלת  $Q$  באמצעות הנוסחה:

$$Q(s, a) = Q(s, a) + \alpha(R + \gamma Q(s', a') - Q(s, a))$$



# פסאודו קוד SARSA

Sarsa (on-policy TD control) for estimating  $Q \approx q_*$

Initialize  $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$ , arbitrarily, and  $Q(\text{terminal-state}, \cdot) = 0$

Repeat (for each episode):

Initialize  $S$

Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)

Repeat (for each step of episode):

Take action  $A$ , observe  $R, S'$

Choose  $A'$  from  $S'$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)

$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma Q(S', A') - Q(S, A)]$

$S \leftarrow S'; A \leftarrow A';$

until  $S$  is terminal

# אלגוריתם Q-learning

- הסוכן נמצא במצב  $S$  יבחר את הפעולה  $A$  בהתאם לטבלת  $Q$  ולמדיניות  $\epsilon$ -greedy.
- הסוכן ידגום את הסביבה ויקבל את המצב הבא:  $S', A, R, S'$ .
- בחירת  $A'$  תעשה לפי טבלת  $Q$  בלבד ללא  $\epsilon$ -greedy. כלומר נבחר את הפעולה שתביא את הערך המירבי  $\max_{a'} Q(s', a')$ .
- אחרי כל צעד ישתמש הסוכן בנתונים כדי לעדכן את טבלת  $Q$  באמצעות הנוסחה:

$$Q(s, a) = Q(s, a) + \alpha(R + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a))$$



# Q-Learning

אלגוריתם מאוד דומה הנקרא Q-Learning מבוסס על אותו הרעיון, וההבדל היחידי נעוץ במקום בו אנו עושים שימוש ב  $\epsilon$ -greedy.

Q-learning (off-policy TD control) for estimating  $\pi \approx \pi_*$

Initialize  $Q(s, a)$ , for all  $s \in \mathcal{S}, a \in \mathcal{A}(s)$ , arbitrarily, and  $Q(\text{terminal-state}, \cdot) = 0$

Repeat (for each episode):

Initialize  $S$

Repeat (for each step of episode):

Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)

Take action  $A$ , observe  $R, S'$

$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$

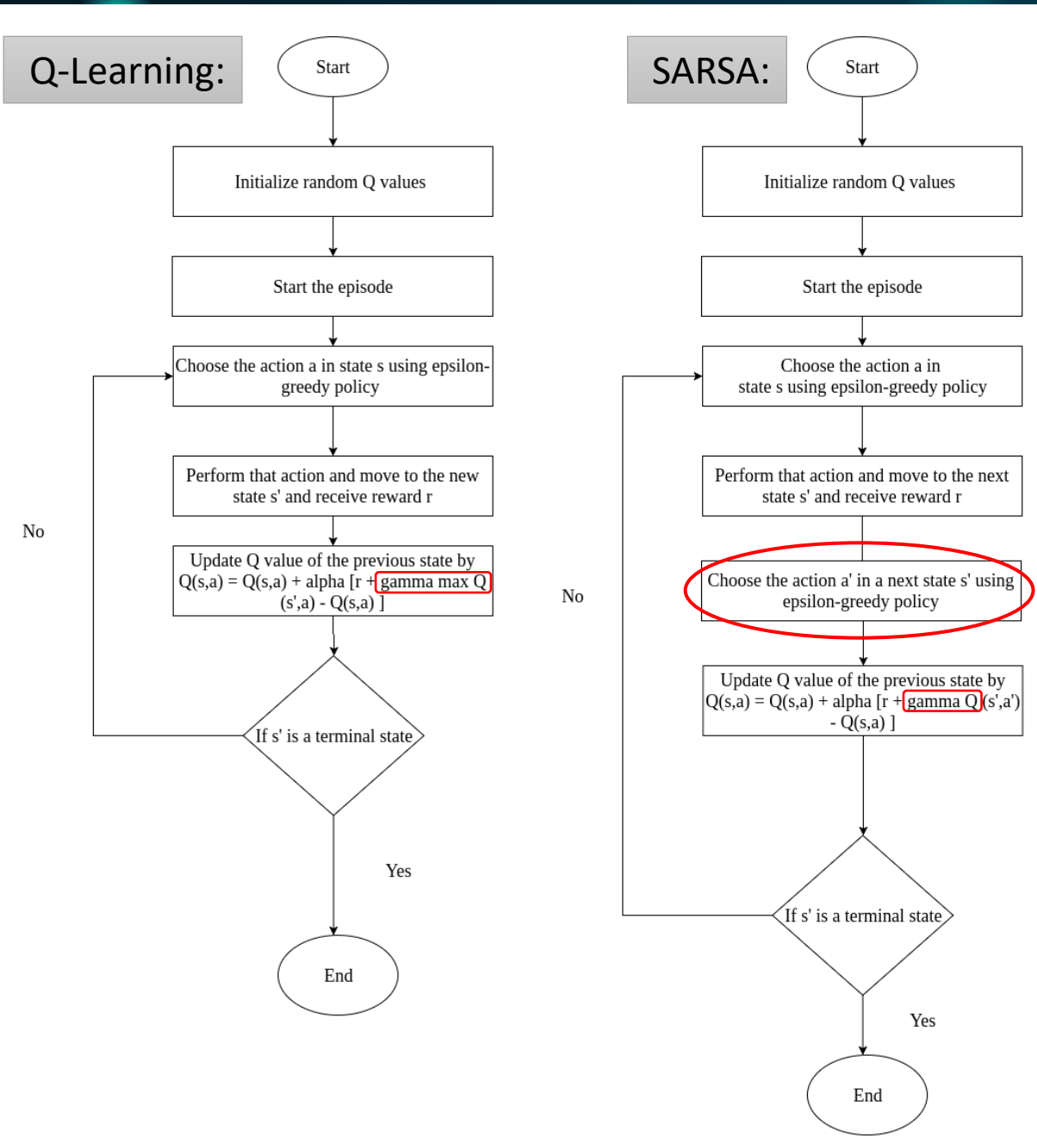
$S \leftarrow S'$

until  $S$  is terminal

# Q-Learning vs SARSA

- כפי שניתן לראות מן התרשים ההבדל היחידי בין האלגוריתמים בבחירת  $a'$ .
- בשניהם בוחרים את  $a$  לפי  $\epsilon$ -greedy
- ב SARSA – בוחרים את  $a'$  תוך שימוש ב  $\epsilon$ -greedy.
- ב Q-learning בוחרים את  $a'$  לפי הערך הגבוה ביותר בטבלת  $Q$  –  $(\text{argmax})$ .

\* Oreilly, hands-on-reinforcement-learning





# n-step TD

• TD(0) – מעדכנים כל צעד

$$V(s) = V(s) + \alpha(R + \gamma V(s') - V(s))$$

• MC – מעדכנים בסוף מהלך לפי חישוב התגמול במהלך זה.

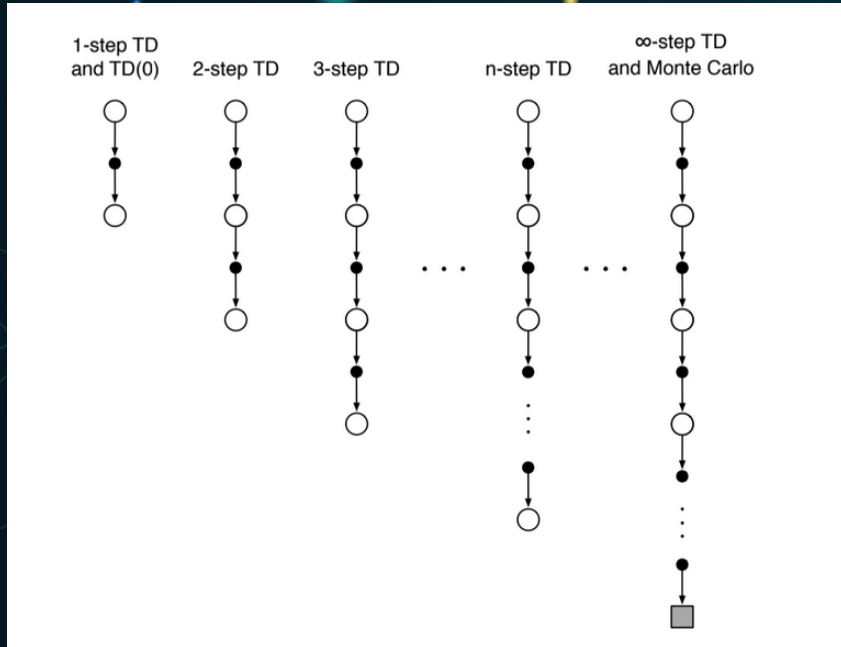
$$G_T = R_0 + \gamma R_1 + \gamma^2 R_2 + \dots + \gamma^T R_T$$

$$V(s) = V(s) + \alpha(G_t - V(s))$$

• *n-step TD* – אנחנו משלבים בין שיניהם:

$$G_n = R_0 + \gamma R_1 + \dots + \gamma^n R_n + \gamma^{n-1} V(s_{n+1})$$

$$V(s) = V(s) + \alpha(G_n - V(s))$$



# הדגמה - משחק איקס עיגול

• נדגים יישום של האלגוריתם SARSA במשחק איקס עיגול כשהשחקן הוא  $X$ .

## תרגיל

- עליכם ליישם את האלגוריתם של Q-learning במשחק איקס עיגול.
- עליכם ליישם את אחד האלגוריתמים על שחקן  $O$ .
- עליכם ליישם את אלגוריתם SARSA באמצעות טבלת afterState (במקום טבלת  $Q$ ). ראו הסברים בסרטון הבא.