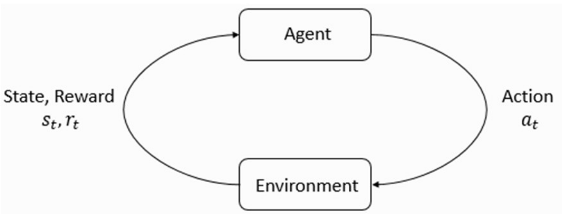
11. Reinforcement Learning (RL)

רוב האלגוריתמים של עולם הלמידה הינם מבוססי דאטה, כלומר, בהינתן מידע מסוים הם מנסים למצוא בו חוקיות מסוימת, ועל בסיסה לבנות מודל שיוכל להתאים למקרים נוספים. אלגוריתמים אלה מחולקים לשניים:

1. אלגוריתמים של למידה מונחית, המבוססים על דאטה $S = \{\boldsymbol{x}, \boldsymbol{y}\}$, כאשר $\boldsymbol{x} \in \mathbb{R}^{n\times d}$ הינו אוסף של אובייקטים (למשל נקודות במרחב, אוסף של תמונות וכדו'), ו-$\boldsymbol{y} \in \mathbb{R}^{n}$ הינו אוסף של labels. לכל אובייקט $x \in \mathbb{R}^{d}$ יש label מתאים $y \in \mathbb{R}^{1}$.
2. אלגוריתמים של למידה לא מונחית עבורם הדאטה $\boldsymbol{x} \in \mathbb{R}^{n\times d}$ הוא אוסף של אובייקטים ללא labels, ומנסים למצוא כללים מסוימים על דאטה זה (למשל – חלוקה לאשכולות, הורדת ממד ועוד).

למידה מבוססת חיזוקים הינה פרדיגמה נוספת תחת התחום של למידת מכונה, כאשר במקרה זה הלמידה לא מסתמכת על דאטה קיים, אלא על חקירה של הסביבה ומציאת המדיניות/האסטרטגיה הטובה ביותר לפעולה. ישנו סוכן שנמצא בסביבה שאינה מוכרת, ועליו לבצע צעדים כך שהתגמול המצטבר אותו הוא יקבל יהיה מקסימלי. בלמידה מבוססת חיזוקים, בניגוד לפרדיגמות האחרות של למידת מכונה, הסביבה לא ידועה מבעוד מועד. הסוכן נמצא באי ודאות ואינו יודע בשום שלב מה הצעד הנכון לעשות, אלא הוא רק מקבל פידבק על הצעדים שלו, וכך הוא לומד מה כדאי לעשות וממה כדאי להימנע. באופן כללי ניתן לומר שמטרת הלמידה היא לייצר אסטרטגיה כך שבכל מיני מצבים לא ידועים הסוכן יבחר בפעולות שבאופן מצטבר יהיו הכי יעילות עבורו. נתאר את תהליך הלמידה באופן גרפי:



איור 11.1 מודל של סוכן וסביבה.

בכל צעד הסוכן נמצא במצב s_t ובוחר פעולה a_t המעבירה אותו למצב s_{t+1}, ובהתאם לכך הוא מקבל מהסביבה תגמול r_t. האופן בה מתבצעת הלמידה היא בעזרת התגמול, כאשר נרצה שהסוכן יבצע פעולות המזכות אותו בתגמול חיובי (-חיזוק) וימנע מפעולות עבורן הוא מקבל תגמול שלילי, ובמצטבר הוא ימקסם את כלל התגמולים עבור כל הצעדים שהוא בחר לעשות. כדי להבין כיצד האלגוריתמים של למידה מבוססת חיזוקים עובדים ראשית יש להגדיר את המושגים השונים, ובנוסף יש לנסח באופן פורמלי את התיאור המתמטי של חלקי הבעיה השונים.

11.1 Introduction to RL

בפרק זה נגדיר באופן פורמלי תהליכי מרקוב, בעזרתם ניתן לתאר בעיות של למידה מבוססת חיזוקים, ונראה כיצד ניתן למצוא אופטימום לבעיות אלו בהינתן מודל וכל הפרמטרים שלו. לאחר מכן נדון בקצרה במספר שיטות המנסות למצוא אסטרטגיה אופטימלית עבור תהליך מרקוב כאשר לא כל הפרמטרים של המודל נתונים, ובפרקים הבאים נדבר על שיטות אלה בהרחבה. שיטות אלה הן למעשה הלב של למידה מבוססת חיזוקים, כיוון שהן מנסות למצוא אסטרטגיה אופטימלית על בסיס תגמולים ללא ידיעת הפרמטרים של המודל המרקובי עבורו רוצים למצוא אופטימום.

11.1.1 Markov Decision Process (MDP) and RL

המודל המתמטי העיקרי עליו בנויים האלגוריתמים השונים של RL הינו תהליך החלטה מרקובי, כלומר תהליך שבה המעברים בין המצבים מקיים את תכונת מרקוב, לפיה ההתפלגות של מצב מסוים תלויה רק במצב הקודם לו:

$$P(s_{t+1} = j|s_1, \dots, s_t) = P(s_{t+1} = j|s_t)$$

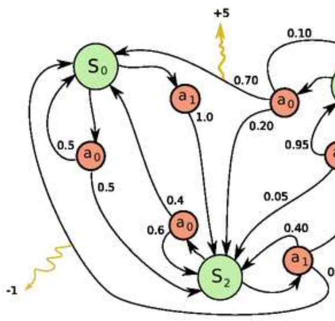
תהליך קבלת החלטות מרקובי מתואר על ידי סט הפרמטרים $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}\}$:

- State space $(\mathcal{S})$ – מרחב המצבים של המערכת. המצב ההתחלתי מסומן ב-S_0.
- Action space $(\mathcal{A})$ – מרחב הפעולות. A_s הוא מרחב הפעולות האפשריות במצב S.
- Transition $(\mathcal{T})$ – הביטוי: $T(s'|s, a) \to [0, 1]$ הינו פונקציית מעבר, המחשבת את ההסתברות לעבור בזמן t ממצב s_t למצב s_{t+1} על ידי הפעולה a: $T(s'|s, a) = \mathcal{P}(s_{t+1} = s'|s_t = s, a_t = a)$. ביטוי זה למעשה מייצג את המודל – מה ההסתברות שבחירת הפעולה a במצב s תביא את הסוכן למצב s'.
- Reward $(\mathcal{R})$ – הביטוי: $\mathcal{R}_a(s, s') \to \mathbb{R}$ הינו פונקציה הנותנת תגמול/רווח לכל פעולה a הגורמת למעבר ממצב s למצב s', כאשר בדרך כלל $\mathcal{R}_a \in [0, 1]$. לעיתים מסמנים את התגמול של הצעד בזמן t ב-r_t.

המרקוביות של התהליך באה לידי ביטוי בכך שמצב s_t מכיל בתוכו את כל המידע הנחוץ בכדי לקבל החלטה לגבי a_t, או במילים אחרות – כל ההיסטוריה בעצם שמורה בתוך המצב s_t.

ריצה של MDP מאופיינת על ידי הרביעייה הסדורה $\{s_t, a_t, r_t, s_{t+1}\}$ – פעולה a_t המתרחשת בזמן t וגורמת למעבר ממצב s_t למצב s_{t+1}, ובנוסף מקבלת תגמול מיידי r_t, כאשר $r_t \sim \mathcal{R}(s_t, a_t)$ ו-$s_{t+1} \sim p(\cdot\,|s_t, a_t)$.

מסלול (trajectory) הינו סט של שלשות $\tau = \{s_0, a_0, r_0, \dots, s_t, a_t, r_t\}$, כאשר המצב התחלתי מוגרל מהתפלגות כלשהיא $s_0 \sim \rho_0(\cdot)$, והמעבר בין המצבים יכול להיות דטרמיניסטי $s_{t+1} = f(s_t, a_t)$ או סטוכסטי $s_{t+1} \sim p(\cdot\,|s_t, a_t)$.



איור 11.2 תהליך קבלת החלטות מרקובי. ישנם שלושה מצבים – $\{s_0, s_1, s_2\}$, ובכל אחד מהם יש שתי פעולות אפשריות (עם הסתברויות מעבר שונות) – $\{a_0, a_1\}$. עבור חלק מהפעולות יש תגמול שונה מ-0. מסלול יהיה מעבר על אוסף של מצבים דרך אוסף של פעולות, שלכל אחד מהן יש תגמול.

אסטרטגיה של סוכן, המסומנת ב-π, הינה בחירה של אוסף מהלכים. בבעיות של למידה מבוססת חיזוקים, נרצה למצוא **אסטרטגיה אופטימלית** (Optimal Policy) $\pi: S \to A$ הממקסמת את **התגמול המצטבר** $\sum_{t=0}^{\infty} \mathcal{R}\big(s_t, \pi(s_t)\big)$. כיוון שלא תמיד אפשרי לחשב באופן ישיר את האסטרטגיה האופטימלית, ניתן להגדיר ערך החזרה (Return) המבטא סכום של תגמולים, ומנסים למקסם את התוחלת שלו $\mathbb{E}[Return|\mathcal{S}, \mathcal{A}]$. ערך ההחזרה הכי נפוץ נקרא discount return, והוא מוגדר באופן הבא: עבור פרמטר $\gamma \in (0,1)$, ה-Return הינו הסכום הבא:

$$Return = \sum_{t=1}^{T} \gamma^{t-1} r_t$$

אם $\gamma = 0$, אז מתעניינים רק בתגמול המיידי, וככל ש-γ גדל כך נותנים יותר משמעות לתגמולים עתידיים. כיוון ש-$r_t \in [0,1]$, הסכום חסום על ידי $\frac{1}{1-\gamma}$.

התוחלת של ערך ההחזרה נקראת Value function, והיא נותנת לכל מצב ערך מסוים המשקף את תוחלת התגמול שניתן להשיג דרך מצב זה. באופן פורמלי, כאשר מתחילים ממצב s, ה-Value function מוגדר להיות:

$$\mathcal{V}^{\pi}(s) = \mathbb{E}[R(\tau)|s_0 = s]$$

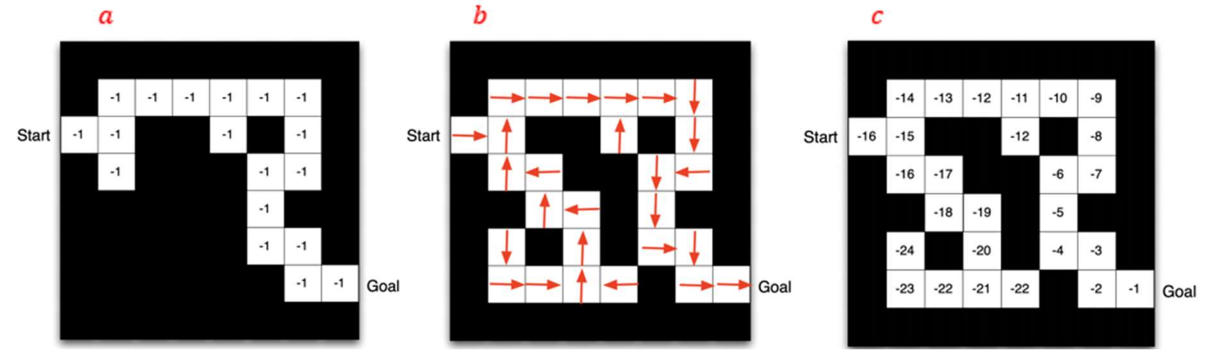
בעזרת ביטוי זה ניתן לחשב את **האסטרטגיה האופטימלית**, כאשר ניתן לנקוט בגישה ישירה ובגישה עקיפה. הגישה הישירה מנסה למצוא בכל מצב מה הפעולה הכי כדאית. בהתאם לכך, חישוב האסטרטגיה האופטימלית יעשה באופן הבא:

$$\pi(s) = \arg\max_a \sum_{s'} p_a(s, s')\big(\mathcal{R}_a(s, s') + \gamma \mathcal{V}(s')\big)$$

לעיתים החישוב הישיר מסובך, כיוון שהוא צריך לקחת בחשבון את כל הפעולות האפשריות, ולכן מסתכלים רק על ה-Value function. לאחר שלכל מצב יש ערך מסוים, בכל מצב הסוכן יעבור למצב בעל הערך הכי גדול מבין כל המצבים האפשריים אליהם ניתן לעבור. חישוב הערך של כל מצב נעשה באופן הבא:

$$\mathcal{V}(s) = \sum_{s'} p_{\pi}(s, s')\big(\mathcal{R}_{\pi}(s, s') + \gamma V(s')\big)$$

ניתן לשים לב שבעוד הגישה הראשונה מתמקדת במציאת **אסטרטגיה/מדיניות אופטימלית** על בסיס הפעולות האפשריות בכל מצב, הגישה השנייה לא מסתכלת על הפעולות אלא על הערך של כל מצב, המשקף את **תוחלת התגמול** שניתן להשיג כאשר נמצאים במצב זה.



איור 11.3 a) מודל: המצב של הסוכן הוא המשבצת בו הוא נמצא, הפעולות האפשריות הן ארבעת הכיוונים, כל פעולה גוררת תגמול של -1, והסתברויות המעבר נקבעות לפי הצבעים של המשבצות (אי אפשר ללכת למשבצות שחורות). b) מדיניות – החלטה בכל מצב איזה צעד לבצע. c) Value של כל משבצת.

לסיכום, ניתן לומר שכל התחום של RL מבוסס על שלוש אבני יסוד:

- מודל: האופן בו אנו מתארים את מרחב המצבים והפעולות. המודל יכול להיות נתון או שנצטרך לשערך אותו, והוא מורכב מהסתברויות מעבר בין מצבים ותגמול עבור כל צעד:

$$\mathcal{P}_{ss'}^{a} = p_{\pi}(s,s') = P(s_{t+1} = s' | s_t = s, a_t = a)$$

$$\mathcal{R}_{ss}^{a} = \mathcal{R}_{\pi}(s,s') = \mathbb{E}\left[r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\right]$$

- Value function – פונקציה המתארת את התוחלת של התגמולים העתידיים:

$$\mathcal{V}^{\pi}(s) = \mathbb{E}[R(\tau)|s_0 = s]$$

- מדיניות/אסטרטגיה (Policy) – בחירה (דטרמיניסטית או אקראית) של צעד בכל מצב נתון:

$$\pi(s|a)$$

11.1.2 Bellman Equation

לאחר שהגדרנו את המטרה של למידה מבוססת חיזוקים, ניתן לדבר על שיטות לחישוב אסטרטגיה אופטימלית. בפרק זה נתייחס למקרה הספציפי בו נתון מודל מרקובי עם כל הפרמטרים שלו, כלומר אוסף המצבים, הפעולות והסתברויות המעבר ידועים. כאמור, Value function הינה התוחלת של ערך ההחזרה עבור אסטרטגיה נתונה π, כאשר מתחילים ממצב s:

$$\mathcal{V}^{\pi}(s) = \mathbb{E}[R(\tau)|s_0 = s] = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right]$$

ביטוי זה מסתכל על הערך של כל מצב, בלי להתייחס לפעולות המעבירות את הסוכן ממצב אחד למצב אחר. נתינת ערך לכל מצב יכולה לסייע במציאת אסטרטגיה אופטימלית, כיוון שהיא מדרגת את המצבים השונים של המודל. באופן דומה, ניתן להגדיר את ה-Action-Value function – התוחלת של ערך ההחזרה עבור אסטרטגיה נתונה π, כאשר במצב s מבצעים את פעולה a, ולאחר מכן ממשיכים לפי האסטרטגיה π:

$$\mathcal{Q}^{\pi}(s,a) = \mathbb{E}[R(\tau)|s_0 = s, a_0 = a] = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = t\right]$$

ביטוי זה מסתכל על הזוג (s_t, a_t), כלומר בכל מצב יש התייחסות למצב הנוכחי ולפעולות האפשריות במצב זה. בדומה ל-Value function, גם ביטוי זה יכול לסייע במציאת אסטרטגיה אופטימלית, כיוון שהוא מדרג עבור כל מצב את הפעולות האפשריות.

נוכל לסמן ב-$\mathcal{V}^*(s)$ ו-$\mathcal{Q}^*(s,a)$ את הערכים של האסטרטגיה האופטימלית π^* – Optimal Value function ו-Optimal Action-Value function. עבור אסטרטגיה זו מתקיים:

$$\mathcal{V}^*(s) = \max_\pi \mathbb{E}[R(\tau)|s_0 = s]\,, \mathcal{Q}^*(s,a) = \max_\pi \mathbb{E}[R(\tau)|s_0 = s, a_0 = a]$$

הרבה פעמים מתעניינים ביחס שבין $\mathcal{V}$ ו-$\mathcal{Q}$, וניתן להיעזר במעברים הבאים:

$$\mathcal{V}^\pi(s) = \mathbb{E}[\mathcal{Q}^\pi(s,a)]$$

$$\mathcal{V}^*(s) = \max_\pi \mathcal{Q}^*(s,a)$$

באופן קומפקטי ניתן לרשום את $\mathcal{V}^*(s)$ כך:

$$\forall s \in S \quad \mathcal{V}^*(s) = \max_\pi \mathcal{V}^\pi(s)$$

כלומר, האסטרטגיה π^* הינה האופטימלית עבור כל מצב s.

כעת נתון מודל מרקובי עם כל הפרמטרים שלו – אוסף המצבים והפעולות, הסתברויות המעבר והתגמול עבור כל פעולה, ומעוניינים למצוא דרך פעולה אופטימלית עבור מודל זה. ניתן לעשות זאת בשתי דרכים עיקריות – מציאת האסטרטגיה $\pi(a|s)$ האופטימלית, או חישוב ה-Value של כל מצב ובחירת מצבים בהתאם לערך זה. משימות אלו יכולות להיות מסובכות מאוד עבור משימות מורכבות וגדולות, ולכן לעיתים קרובות משתמשים בשיטות איטרטיביות ובקירובים על מנת לדעת כיצד לנהוג בכל מצב. הדרך הפשוטה לחישוב $\mathcal{V}^\pi(s)$ משתמשת ב-Bellman equation, המבוססת על תכנות דינמי. נפתח את הביטוי של $\mathcal{V}^\pi(s)$ מתוך ההגדרה שלו:

$$\mathcal{V}^\pi(s) = \mathbb{E}[R(\tau)|s_0 = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right]$$

נפצל את הסכום שבתוחלת לשני איברים – האיבר הראשון ויתר האיברים:

$$= \mathbb{E}_\pi\left[r_{t+1} + \gamma \cdot \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s\right]$$

כעת נשתמש בהגדרת התוחלת ונקבל:

$$= \sum_{a,s'} \pi(a|s) p_\pi(s,s') \left(\mathcal{R}_\pi(s,s') + \gamma \cdot \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s\right]\right)$$

$$= \sum_{a,s'} \pi(a|s) p_\pi(s,s') \left(\mathcal{R}_\pi(s,s') + \gamma \cdot \mathcal{V}^\pi(s')\right)$$

הביטוי המתקבל הוא מערכת משוואות לינאריות הניתנות לפתרון באופן אנליטי, אם כי סיבוכיות החישוב יקרה. נסמן:

$$V = [V_1, \dots, V_n]^T, R = [r_1, \dots, r_n]^T$$

$$T = \begin{pmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nn} \end{pmatrix}$$

ונקבל משוואה מטריציונית:

$$V = R + \gamma T V \to V = R + \gamma T V$$

$$\to \mathcal{V}^\pi(s) = (\mathbb{I}_n - \gamma T)^{-1} R$$

בגלל שהערכים העצמיים של T חסומים על ידי 1, בהכרח יהיה ניתן להפוך את $\mathbb{I}_n - \gamma T$ מה שמבטיח שיהיה פתרון למשוואה, ופתרון זה הוא אף יחיד עבור $\mathcal{V}^\pi$. כשמוצאים את V ניתן למצוא גם את $\mathcal{Q}^\pi$ על ידי הקשר:

$$\mathcal{Q}^\pi(s,a) = \sum_{s'} p_\pi(s,s')\left(\mathcal{R}_\pi(s,s') + \gamma \mathcal{V}^\pi(s')\right) = \sum_{s'} p_\pi(s,s')\left(\mathcal{R}_\pi(s,s') + \gamma \sum_{a'} \pi(a'|s')\, \mathcal{Q}^\pi(a'|s')\right)$$

Iterative Policy Evaluation

הסיבוכיות של היפוך מטריצה הינו $\mathcal{O}(n^3)$, ועבור n גדול החישוב נהיה מאוד יקר ולא יעיל. כדי לחשב את הפתרון באופן יעיל, ניתן כאמור להשתמש בשיטות איטרטיביות. שיטות אלו מבוססות על אופרטור בלמן, המוגדר באופן הבא:

$$BO(V) = R^\pi + \gamma T^\pi \cdot V$$

ניתן להוכיח שאופרטור זה הינו העתקה מכווצת (contractive mapping), כלומר הוא מקיים את התנאי:

$$\forall x, y: \|f(x) - f(y)\| < \gamma \|x - y\| \; for \; 0 < \gamma < 1$$

במילים: עבור שני וקטורים במרחב, אופרטור $f(\cdot)$ ומספר γ החסום בין 0 ל-1, אם נפעיל את האופרטור על כל אחד מהווקטורים ונחשב את נורמת ההפרש, נקבל מספר **קטן** יותר מאשר הנורמה בין הווקטורים כפול הפקטור γ. אופרטור המקיים תכונה זו הינו העתקה מכווצת, כיוון שנורמת ההפרש של האופרטור על שני וקטורים קטנה מנורמת ההפרש בין הווקטורים עצמם. הוכחה:

$$\|f(u) - f(v)\|_\infty = \|R^\pi + \gamma T^\pi \cdot v - (R^\pi + \gamma T^\pi \cdot u)\|_\infty = \|\gamma T^\pi (v - u)\|_\infty$$

מטריקת אינסוף מוגדרת לפי: $\|u - v\|_\infty = \max_{s \in \mathcal{S}} |u(s) - v(s)|$. לכן נוכל לרשום:

$$\|\gamma T^\pi (v - u)\|_\infty \leq \|\gamma T^\pi \mathbb{1} \|u - v\|_\infty\|_\infty$$

הביטוי $T^\pi \mathbb{1}$ למעשה סוכם את כל ערכי מטריצת המעברים, לכן הוא מסתכם ל-1, ונקבל:

$$= \gamma \|u - v\|_\infty$$

ובכך הוכחנו את הדרוש.

לפי משפט נקודת השבת של בנך, להעתקה מכווצת יש נקודת שבת (fixed point) יחידה המקיימת $x = f(x)$ וסדרה $x_{t+1} = f(x_t)$ המתכנסת לאותה נקודת שבת. לכן נוכל להשתמש באלגוריתם איטרטיבי עבור $\mathcal{V}^\pi$ שיביא אותנו לנקודת שבת, ולפי המשפט זוהי נקודת השבת היחידה וממילא הגענו להתכנסות. בפועל, נשתמש באלגוריתם האיטרטיבי הבא:

$$V_{k+1} = BO(V_k) = R^\pi + \gamma T^\pi \cdot V_k$$

נסתכל על הדוגמא הבאה:

$$T^\pi = \begin{pmatrix} 0.8 & 0.1 & 0.1 & 0 & 0 \\ 0.1 & 0.8 & 0.1 & 0 & 0 \\ 0 & 0.1 & 0.8 & 0.1 & 0 \\ 0 & 0 & 0.1 & 0.8 & 0.1 \\ 0 & 0 & 0.1 & 0.1 & 0.8 \end{pmatrix}, \mathcal{R}^\pi = \begin{pmatrix} 0.1 \\ 1.3 \\ 3.4 \\ 1.9 \\ 0.4 \end{pmatrix}, \gamma = 0.9$$

באמצעות השיטה האיטרטיבית נקבל:

$$V_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, V_1 = \begin{pmatrix} 0.1 \\ 1.3 \\ 3.4 \\ 1.9 \\ 0.4 \end{pmatrix}, V_2 = \begin{pmatrix} 0.6 \\ 2.6 \\ 6.1 \\ 3.7 \\ 1.2 \end{pmatrix}, \dots, V_{10} = \begin{pmatrix} 7.6 \\ 10.8 \\ 18.2 \\ 16.0 \\ 9.8 \end{pmatrix}, \dots, V_{50} = \begin{pmatrix} 14.5 \\ 17.1 \\ 26.4 \\ 26.8 \\ 18.4 \end{pmatrix}, V^\pi = \begin{pmatrix} 14.7 \\ 17.9 \\ 26.6 \\ 27.1 \\ 18.7 \end{pmatrix}$$

ניתן לשים לב שאחרי 50 איטרציות הפתרון המתקבל בצורה האיטרטיבית קרוב מאוד לפתרון המתקבל בצורה האנליטית.

Policy Iteration (PI)

חישוב ה-Value function מאפשר לנו לחשב את ערכו של $\mathcal{V}^\pi(s)$ עבור כל s, אך הוא אינו מבטיח שנגיע לאסטרטגיה האופטימלית. נניח והצלחנו לחשב את $\mathcal{V}^\pi(s)$ וממנו אנו יודעים לגזור אסטרטגיה, עדיין יתכן שקיימת פעולה a שיותר משתלמת מאשר הפעולה המוצעת לפי האסטרטגיה הנגזרת מ-$\mathcal{V}^\pi(s)$. באופן פורמלי ניתן לתאר זאת בצורה פשוטה – נניח שחישבנו את $\mathcal{V}^\pi(s)$ ואת $\mathcal{Q}^\pi(s,a)$ יתכן וקיימת פעולה עבורה:

$$\text{for such } s, a: \quad \mathcal{Q}^{\pi}(s,a) > \mathcal{V}^{\pi}(s)$$

אם קיימת פעולה כזו, אז ישתלם לבחור בה ורק לאחר מכן לחזור לפעול בהתאם לאסטרטגיה $\pi(a|s)$ הנגזרת מחישוב ה-Value function. למעשה, ניתן לחפש את **כל** הפעולות עבורן כדאי לבצע פעולה מסיימת עבורה התגמול יהיה גבוה יותר מאשר האסטרטגיה של $\mathcal{V}^{\pi}(s)$. באופן פורמלי יותר, נרצה להגדיר אסטרטגיה דטרמיניסטית, עבורה בהסתברות 1 ננקוט בכל מצב s בפעולה הכי כדאית a:

$$\pi'(a|s) = 1 \text{ for } a = \arg\max_{a'} \mathcal{Q}^{\pi}(s,a')$$

נשים לב שרעיון זה הוא בעצם להשתמש באסטרטגיה גרידית – בכל מצב לנקוט בפעולה הכי משתלמת בטווח של צעד יחיד. השאלה העולה היא כמובן – מדוע זה בהכרח נכון? כלומר, האם הרעיון של לבחור **בכל** צעד את a האופטימלי בהכרח תוביל לקבלת אסטרטגיה אופטימלית עבור כל הצעדים כולם יחד? בכדי להוכיח זאת ננסח זאת כמשפט:

בהינתן 2 אסטרטגיות π, π', כאשר π' דטרמיניסטית, אז כאשר $\mathcal{Q}^{\pi}(s,\pi'(s)) > \mathcal{V}^{\pi}(s)$ בהכרח לכל s יתקיים: $\mathcal{V}^{\pi'}(s) > \mathcal{V}^{\pi}(s)$. ראשית נפתח לפי הגדרה:

$$\mathcal{V}^{\pi}(s) < \mathcal{Q}^{\pi}(s,\pi'(s)) = \mathbb{E}_{\pi}\left[r_{t+1} + \gamma \cdot \mathcal{V}^{\pi}(s_{t+1}) | s_t = s, a_t = \pi'(s)\right]$$

כיוון שהאסטרטגיה הינה דטרמיניסטית, הפעולה הנבחרת אינה רנדומלית ביחס ל-π', ולכן נוכל לרשום:

$$= \mathbb{E}_{\pi'}[r_{t+1} + \gamma \cdot \mathcal{V}^{\pi}(s_{t+1}) | s_t = s]$$

כעת לפי אותו אי שוויון שבהנחה נוכל לבצע את אותו חישוב גם לצעד הבא s_{t+2}:

$$< \mathbb{E}_{\pi'}\left[r_{t+1} + \gamma \cdot \mathcal{Q}^{\pi}\left(s_{t+1}, \pi'(s_{t+1})\right) | s_t = s\right]$$

וזה שוב שווה ל:

$$= \mathbb{E}_{\pi'}[r_{t+1} + \gamma \cdot r_{t+2} + \gamma^2 \cdot \mathcal{V}^{\pi}(s_{t+2}) | s_t = s]$$

וכך הלאה, ולמעשה הוכחנו את הדרוש – נקיטת הפעולה הכי יעילה בכל מצב תמיד תהיה יותר טובה מהפתרון של $\mathcal{V}^{\pi}(s)$. כעת יש בידינו שתי טכניקות שאנו יודעים לבצע:

Evaluation (E) – בהינתן אסטרטגיה מסוימת נוכל לפתור את משוואות בלמן ולקבל את $\mathcal{V}^{\pi}(s)$ ו-$\mathcal{Q}^{\pi}(s,a)$.

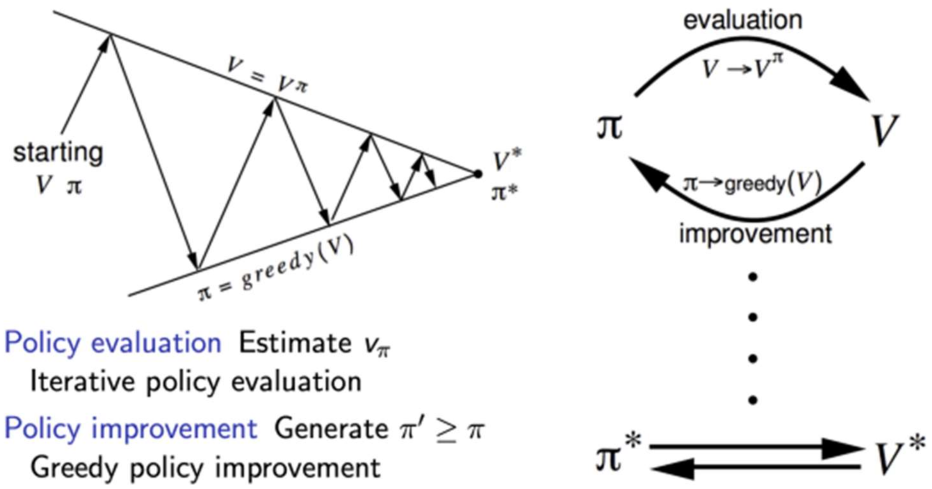
Improve (I) – בהינתן הערך של value function,

ניתן להתחיל מאסטרטגיה רנדומלית, ואז לבצע איטרציות המורכבות משתי הטכניקות האלה באופן הבא:

$$\pi_0 \xrightarrow{E} \mathcal{V}^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} \mathcal{V}^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots$$

תהליך זה נקרא Policy iteration – בכל צעד בו יש לנו אסטרטגיה נפתור עבורה משוואת בלמן ובכך נחשב את ה-Value function שלה, ולאחר מכן נשפר את האסטרטגיה באמצעות policy improvement, שכאמור מבצע בחירה גרידית שבטווח הקצר טובה יותר מאשר ה-Value function שחישבנו. ניתן להוכיח שאחרי מספר סופי של איטרציות האסטרטגיה תתכנס לנקודת שבת (fixed point), ואז הפעולה הבאה לפי האסטרטגיה תהיה זהה לבחירה הגרידית:

$$\pi(s) = \arg\max_{a} \mathcal{Q}^{\pi}(s,a) = \pi'(s)$$



איור 11.4 Policy iteration – ביצוע איטרציות של Policy evaluation ו-Policy improvement על מנת למצוא בכל שלב את ה- value function ולשפר אותו באמצעות בחירה גרידית.

Bellman optimality equations

השלב הבא בשימוש ב-Policy iteration הוא **להוכיח** שהאסטרטגיה אליה מתכנסים הינה אופטימלית. נסמן את נקודת השבת ב-π^* ונקבל את הקשר הבא:

$$\mathcal{V}^{\pi^*}(s) \equiv \mathcal{V}^*(s) = \max_a \mathcal{Q}^*(s,a) = \max_a \sum_{s'} p_\pi(s,s')\Big(\mathcal{R}_\pi(s,s') + \gamma \cdot \mathcal{V}^*(s')\Big)$$

ובאופן דומה:

$$\mathcal{Q}^*(s,a) = \sum_{s'} p_\pi(s,s')\left(\mathcal{R}_\pi(s,s') + \gamma \cdot \max_{a'} \mathcal{Q}^*(s',a')\right)$$

משוואות אלה נקראות Bellman optimality equation. ניתן לשים לב שהן מאוד דומות למשוואות בלמן מהן יצאנו, אך במקום התוחלת שהייתה לנו בהתחלה, כעת יש max. נרצה להראות שהפתרון של משוואות אלה הוא ה-Value של **האסטרטגיה האופטימלית**. ננסח את הטענה באופן הבא:

אסטרטגיה הינה אופטימלית אם ורק אם היא מקיימת את Bellman optimality equation. כיוון אחד להוכחה הוא טריוויאלי – אם האסטרטגיה הינה אופטימלית אז היא בהכרח מקיימת את משוואות האופטימליות, כיוון שהראינו שהן מתקבלות מנקודת השבת אליה האיטרציות מתכנסות. אם האסטרטגיה לא הייתה אופטימלית אז היה ניתן לשפר עוד את האסטרטגיה ולא היינו מגיעים עדיין לנקודת השבת. בשביל להוכיח את הכיוון השני נשתמש שוב ברעיון של העתקה מכווצת. נגדיר את האופרטור הבא:

$$BV(s) = \max_a \sum_{s'} p_\pi(s,s')\big(\mathcal{R}_\pi(s,s') + \gamma \cdot \mathcal{V}(s)\big)$$

ניתן להראות שאופרטור זה הינו העתקה מכווצת, וממילא לפי המשפט של בנך יש לו נקודת שבת יחידה. כיוון שהראינו ששימוש ב-Policy iteration מביא את האסטרטגיה לנקודת שבת מסוימת, נוכל לצרף לכך את העובדה שהאופרטור שהגדרנו הינו העתקה מכווצת וממילא נקבל שאותה נקודת שבת הינה יחידה, וממילא אופטימלית.

Value Iteration

הראנו שבעזרת שיטת Policy iteration ניתן להגיע לאסטרטגיה אופטימלית, אך התהליך יכול להיות איטי. ניתן לנקוט גם בגישה יותר ישירה ולנסות לחשב באופן ישיר את הפתרון של משוואות האופטימליות של בלמן (ופתרון הינו אופטימלי כיוון שהראינו שהפתרון הוא נקודת שבת יחידה). נתחיל עם פתרון רנדומלי $\mathcal{V}_0$ ולאחר מכן נצבע איטרציות באופן הבא עד שנגיע להתכנסות:

$$\mathcal{V}_{k+1} = \max_a \sum_{s'} p_\pi(s,s')\big(\mathcal{R}_\pi(s,s') + \gamma \cdot \mathcal{V}_k(s')\big)$$

נשים לב שבשיטה זו אין לנו מידע לגבי האסטרטגיה אלא רק חישבנו את ה-Value function, אך ממנה ניתן לגזור את Q ואז לבחור באסטרטגיה גרידית, שהינה במקרה זה גם אופטימלית:

$$\pi(s) = \arg\max_a \mathcal{Q}^\pi(s,a)$$

ניתן להראות כי בשיטה זו ההתכנסות מהירה יותר ודרושות פחות איטרציות מהשיטה הקודמת, אך כל איטרציה יותר מורכבת.

Limitations

לשתי השיטות – Policy iteration ו-Value iteration – יש שני חסרונות מרכזיים:

1. הן דורשות לדעת את המודל והסביבה באופן שלם ומדויק.

2. הן דורשות לעדכן בכל שלב את כל המצבים בו זמנית. עבור מערכות עם הרבה מצבים, זה לא מעשי.

11.1.3 Learning Algorithms

בפרק הקודם הוסבר כיצד ניתן לחשב את האסטרטגיה האופטימלית וערך ההחזרה **בהינתן** מודל מרקובי. השתמשנו בשתי הנחות עיקריות על מנת להתמודד עם הבעיה:

1. Tabular MDP – הנחנו שהבעיה סופית ולא גדולה מדי, כך שנוכל לייצג אותה בזיכרון ולפתור אותה.

2. Known environment – הנחנו שהמודל ידוע לנו, כלומר נתונה לנו מטריצת המעברים שקובעת מה הסיכוי לעבור ממצב s למצב s' כשנוקטים בפעולה a (סימנו את זה בתור $\mathcal{P}_{ss'}^a = p_\pi(s,s')$), ובנוסף נתון לנו מה ה-reward המתקבל עבור כל action (סימנו את זה בתור $\mathcal{R}_{ss'}^a = \mathcal{R}_\pi(s,s')$).

בעזרת שתי ההנחות פיתחנו את משוואות בלמן, כאשר היו לנו שני צמדים של משוואות. משוואות בלמן עבור אסטרטגיה נתונה נכתבות באופן הבא:

$$\mathcal{V}^\pi(s) = \sum_{a,s'} \pi(a|s)\mathcal{P}_{ss'}^a\left(\mathcal{R}_{ss'}^a + \gamma \cdot \mathcal{V}^\pi(s')\right)$$

$$\mathcal{Q}^\pi(s,a) = \sum_{s'} \mathcal{P}_{ss'}^a\left(\mathcal{R}_{ss'}^a + \gamma \sum_{a'} \mathcal{Q}^\pi(s',a')\right)$$

ובנוסף פיתחנו את המשוואות עבור הפתרון האופטימלי:

$$\mathcal{V}^*(s) = \max_a \sum_{s'} \mathcal{P}_{ss'}^a\big(\mathcal{R}_{ss'}^a + \gamma \cdot \mathcal{V}^*(s')\big)$$

$$\mathcal{Q}^*(s,a) = \sum_{s'} \mathcal{P}_{ss'}^a\left(\mathcal{R}_{ss}^a + \gamma \max_a \mathcal{Q}^*(s',a')\right)$$

הראינו שתי דרכים להגיע לפתרון האופטימלי:

1. Policy iteration המורכב מ-Policy evaluation ולאחריו Policy improvement.

2. Value iteration – פתרון משוואות בלמן באופן ישיר בעזרת איטרציות על ה-Value function.

כאמור, דרכי פתרון אלו מניחים שהמודל ידוע, ובנוסף שמרחב המצבים אינו גדול מדי ויכול להיות מיוצג בזיכרון. האתגר האמיתי מתחיל בנקודה בה לפחות אחת מהנחות אלה אינה תקפה, ולמעשה פה מתחיל התפקיד של אלגוריתמי RL. עיקר ההתמקדות של אלגוריתמים אלו יהיה למצוא באופן יעיל את האסטרטגיה האופטימלית כאשר לא נתונים הפרמטרים של המודל, ואז צריך לשערך אותם (Model-based learning) או למצוא דרך אחרת לחישוב האסטרטגיה האופטימלית ללא שימוש במודל (Model free learning). אם למשל יש משחק בין משתמש לבין המחשב, אלגוריתמים השייכים ל-Model based learning ינסו ללמוד את המודל של המשחק או להשתמש במודל

קיים, ובעזרת המודל הם ינסו לבחון כיצד יגיב המשתמש לכל תור שהמחשב יבחר. לעומת זאת אלגוריתמים מסוג Model free learning לא יתעניינו בכך, אלא ינסו ללמוד ישירות את האסטרטגיה הטובה ביותר עבור המחשב.

היתרון המשמעותי של אלגוריתמים המסתכלים על המודל של הבעיה (Model-based) נובע מהיכולת לתכנן מספר צעדים קדימה, כאשר עבור כל בחירה של פעולה המודל בוחן את התגובות האפשריות, את הפעולות המתאימות לכל תגובה, וכך הלאה. דוגמא מפורסמת לכך היא תוכנת המחשב AlphaZero שאומנה לשחק משחקי לוח כגון שחמט או גו. במקרים אלו המודל הוא המשחק והחוקים שלו, והתוכנה משתמשת בידע הזה בכדי לבחון את כל הפעולות והתגובות למשך מספר צעדים רב ובחירה של הצעד הטוב ביותר.

עם זאת, בדרך כלל אף בשלב האימון אין לסוכן מידע חיצוני מהו הצעד הנכון באופן אולטימטיבי, ועליו ללמוד רק מהניסיון. עובדה זו מציבה כמה אתגרים, כאשר העיקרי ביניהם הוא הסכנה שהאסטרטגיה הנלמדת תהיה טובה רק עבור המקרים אותם ראה הסוכן, אך לא תתאים למקרים חדשים שיבואו. אלגוריתמים שמחפשים באופן ישיר את האסטרטגיה האופטימלית אמנם לא משתמשים בידע שיכול להגיע מבחינת צעדים עתידיים, אך הם הרבה יותר פשוטים למימוש ולאימון.

באופן מעט יותר פורמלי ניתן לנסח את ההבדל בין הגישות כך: גישת Model-based learning מנסה למצוא את הפרמטרים המגדירים את המודל $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}\}$ ואז בעזרתם לחשב את האסטרטגיה האופטימלית (למשל בעזרת משוואות בלמן). הגישה השנייה לעומת זאת לא מעוניינת לחשב במפורש את הפרמטרים של המודל אלא למצוא באופן ישיר את האסטרטגיה האופטימלית $\pi(a_t|s_t)$ שעבור כל מצב קובעת באיזה פעולה לנקוט. ההבדל בין הגישות נוגע גם לפונקציית המחיר לה נרצה למצוא אופטימום.

בכל אחד משני סוגי הלמידה יש אלגוריתמים שונים, כאשר הם נבדלים אחד מהשני בשאלה מהו האובייקט אותו מעוניינים ללמוד.

Model-free learning

בגישה זו יש שתי קטגוריות מרכזיות של אלגוריתמים:

א. Policy Optimization – ניסוח האסטרטגיה כבעיית אופטימיזציה של מציאת סט הפרמטרים θ הממקסם את $\pi_\theta(a|s)$. פתרון בעיה זו יכול להיעשות באופן ישיר על ידי שיטת Gradient Ascent עבור פונקציית המחיר $\mathcal{J}(\pi_\theta) = \mathbb{E}[R(\tau)]$, או בעזרת קירוב פונקציה זו ומציאת מקסימום עבורה.
ב. Q-learning – שערוך $Q^*(s, a)$ על ידי $Q_\theta(s, a)$. מציאת המשערך האופטימלי יכולה להתבצע על ידי חיפוש θ שיספק את השערוך הטוב ביותר שניתן למצוא, או על ידי מציאת הפעולה שתמקסם את המשערך: $a(s) = \arg\max_a Q_\theta(s, a)$.

השיטות המנסות למצוא אופטימום לאסטרטגיה הן לרוב on-policy, כלומר כל פעולה נקבעת על בסיס האסטרטגיה המעודכנת לפי הפעולה הקודמת. Q-learning לעומת זאת הוא לרוב אלגוריתם off-policy, כלומר בכל פעולה ניתן להשתמש בכל המידע שנצבר עד כה. היתרון של שיטות האופטימיזציה נובע מכך שהן מנסות למצוא באופן ישיר את האסטרטגיה הטובה ביותר, בעוד שאלגוריתם Q-learning רק משערך את $Q^*(s, a)$, ולעיתים השערוך לא מספיק ואז התוצאה המתקבלת אינה מספיק טובה. מצד שני, כאשר השערוך מוצלח, הביצועים של Q-learning טובים יותר, כיוון שהשימוש במידע על העבר מנוצל בצורה יעילה יותר מאשר באלגוריתמים המבצעים אופטימיזציה של האסטרטגיה. שתי הגישות האלה אינן זרות לחלוטין, וישנם אלגוריתמים שמנסים לשלב בין הרעיונות ולנצל את החוזקות והיתרונות שיש לכל גישה.

Model-based learning

גם בגישה זו יש שתי קטגוריות מרכזיות של אלגוריתמים:

א. Model-based RL with a learned model – אלגוריתמים המנסים ללמוד הן את המודל עצמו והן את ה-Value function או את האסטרטגיה π.
ב. Model-based RL with a known model – אלגוריתמים המנסים למצוא את ה-Value function ו/או את האסטרטגיה כאשר המודל עצמו נתון.

ההבדל בין הקטגוריות טמון באתגר איתו מנסים להתמודד. במקרים בהם המודל ידוע, הממד של אי הוודאות לא קיים, ולכן ניתן להתמקד בביצועים אסימפטומטיים. במקרים בהם המודל אינו ידוע, הדגש העיקרי הוא על למידת המודל.

11.2 Model Free Prediction

לאחר שסקרנו בפרק המבוא את הבסיס המתמטי של בעיות RL והצגנו את משוואות בלמן ופתרונן, בפרקים הבאים נציג גישות שונות להתמודדות עם בעיות RL עבורן פתרונות אלה אינן מספיקים – או מפני שהמודל אינו ידוע או מפני שהן ב-Scale גדול יותר מזה שניתן לפתור באמצעות משוואות בלמן. בפרק זה נציג שתי שיטות הבאות להתמודד עם מקרים בהם המודל אינו ידוע (כלומר האלגוריתם הינו Model-Free), והדרך שלהן להתמודד עם אתגר זה הינו **לשערך** את המודל.

11.1.1 Monte-Carlo (MC) Policy Evaluation

האלגוריתם הראשון אותו נציג הינו Monte Carlo, והוא מציע דרך לשערך את ה-Value function בלי לדעת את המודל. ראשית נסביר בקצרה מהו אלגוריתם Monte Carlo ואז נראה כיצד ניתן ליישם אותו בבעיות RL.

נניח ונרצה לשערך תוחלת של פונקציית התפלגות כלשהיא – $\mathbb{E}_p[f(x)]$. התוחלת יכולה להיות סכום או אינטגרל שקשה מאוד לחשב. ניתן לשערך את התוחלת על ידי דגימות רנדומליות מההתפלגות וחישוב הממוצע של הדגימות:

$$x_1, \dots x_n \sim p(x)$$

$$\mathbb{E}_p[f(x)] \approx \frac{1}{n}\sum_i f(x_i)$$

לפי חוק המספרים הגדולים הממוצע של הדגימות מתכנס לתוחלת. משערך זה הינו חסר הטיה, ובנוסף השונות שלה קטנה ביחס לינארי לכמות הדגימות:

$$\mathbb{E}\left[\frac{1}{n}\sum_i f(x_i)\right] = \mathbb{E}[f(x)]$$

$$Var\left[\frac{1}{n}\sum_i f(x_i)\right] = \frac{Var[f(x)]}{n}$$

כאמור לעיל, ה-Value function הינה תוחלת עבור אסטרטגיה נתונה π, כאשר מתחילים ממצב s:

$$\mathcal{V}^\pi(s) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right]$$

אמנם התוחלת היא על סכום אינסופי, אך עבור מקרים רבים נוכל להניח שהריצה היא סופית (Episodic MDP). בפועל נבצע את הפעולה הבאה: עבור אסטרטגיה נתונה π, נייצר ממנה ריצה של T צעדים, ואז ניקח מצב מסוים ונסתכל על כל ה-rewards שמתקבלים בריצה זו החל ממנו. באופן הזה קיבלנו value עבור הערך של אותו מצב. חזרה על אותה פעולה שוב ושוב תייצר ריצות שונות וממילא ערכים שונים למצב מסוים, ומיצוע על פני הערכים ייתן לנו שערוך לערך האמיתי של אותו מצב. נעיר בהערת אגב שכיוון שמצבים יכולים לחזור על עצמם, נוצרת בעיה שבריצה כזו המצבים אינם בלתי תלויים. בכדי להתגבר על כך, אם מצב חוזר על עצמו יותר מפעם אחת מאפשרים לדגום רק את המופע הראשון של אותו מצב ולא את יתר המופעים (ישנן עוד דרכים להתגבר על כך, אך זוהי הדרך פשוטה ביותר). באופן פורמלי, נניח ויש לנו ריצה של T צעדים:

$$S_1, A_1, R_1, \dots, S_{T-1}, A_{T-1}, R_{T-1}, S_T$$

אז דגימה אחת מתוך ההתפלגות $p_\pi(\sum_{k=0}^{\infty} r_{t+k+1} \, | s_t = S_t)$ תראה באופן הבא:

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$$

למשל, הדגימה G_1 תהיה מורכבת מכל ה-rewards שהגיעו לאחר הצעד הראשון: $R_1 + \gamma R_2 + \cdots + \gamma^{T-1} R_T$. הסכום הזה ייתן לנו value עבור אותו מצב ממנו התחלנו (S_1), ועל ידי שערוך אותו מצב שוב ושוב ביחס לריצות שונות נוכל לקבל ערכים שונים, ולאחר מכן למצע אותם בכדי לשערך את ה-value האמיתי של אותו מצב S_1.

באופן פורמלי, העדכון של מצב לאחר כל דגימה נראה כך:

$$\#sample\ of\ s_t: N(S_t) = N(S_t) + 1$$

$$update\ the\ value\ of\ s_t: \mathcal{V}(s_t) = \mathcal{V}(s_t) + \frac{1}{N(S_t)}\big(G_t - \mathcal{V}(s_t)\big)$$

ולאחר הרבה דגימות השערוך מתקבל על ידי התוחלת שלהן:

$$\mathcal{V}(s) = \mathbb{E}_\pi[G_t | s_t = s]$$

באופן סכמתי ניתן לתאר את האלגוריתם באופן הבא:

First-visit MC prediction, for estimating $V \approx v_\pi$

Initialize:
  $\pi \leftarrow$ policy to be evaluated
  $V \leftarrow$ an arbitrary state-value function
  $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:
  Generate an episode using π
  For each state s appearing in the episode:
    $G \leftarrow$ the return that follows the first occurrence of s
    Append G to $Returns(s)$
    $V(s) \leftarrow$ average$(Returns(s))$

איור 11.5 אלגוריתם MC עבור שערוך ה-Value function בהינתן Policy. עבור כל מצב מאתחלים רשימה ריקה, ולאחר מכן מייצרים המון ריצות שונות. עבור כל ריצה, עוברים על כל המצבים ובודקים מה ה-G שלהם, ומוסיפים אותו לרשימה של אותו מצב. לבסוף מחשבים את ה-value של כל מצב על ידי מיצוע הרשימה (=ערכי G השונים) של אותו מצב.

יש מספר יתרונות לשיטה זו: היא מספקת משערך חסר הטיה עבור ה-Value function, מבטיחה התכנסות אחרי מספיק איטרציות, ובנוסף ניתן לשערך באמצעותה את השגיאה. אפשר גם באותה דרך לשערך גם את $\mathcal{Q}^\pi(s,a)$ אך זה יהיה יותר רועש וידרוש יותר דגימות. מן הצד השני יש גם חסרונות לשיטה זו: ראשית, היא מתאימה רק ל-Episodic MDP, אך זו בעיה קלה כיוון שעבור בעיה אינסופית ניתן לקחת ריצה סופית ולחסום את השגיאה באמצעות γ. שנית, ההתכנסות יכולה להיות מאוד איטית, ובנוסף השונות יחסית גבוהה.

11.1.2 Temporal Difference (TD) – Bootstrapping

במקום להסתכל על ריצה שלמה, ניתן אחרי כל צעד לעדכן את ה-Value. ממשוואת בלמן ניתן להראות שהביטוי $R_{t+1} + \gamma\mathcal{V}^\pi(S_{t+1})$ הינו משערך חסר הטיה עבור $\mathcal{V}^\pi(S_t)$. אי אפשר להשתמש במשערך זה כמו שהוא, כיוון שאנחנו לא יודעים את ה-Value function, וממילא הביטוי $\mathcal{V}^\pi(S_{t+1})$ לא ידוע. בשביל בכל זאת לשערך את $\mathcal{V}^\pi(S)$ באמצעות אותו משערך, ניתן להחליף את $\mathcal{V}^\pi(S_{t+1})$ ב-$\mathcal{V}(S_{t+1})$, ולבצע את השערוך באופן הבא:

$$\mathcal{V}^\pi(S_t) = R_{t+1} + \gamma\mathcal{V}(S_{t+1})$$

הרעיון מאחורי השימוש הזה הוא להיעזר במידע שיש לנו מ-R_t. נניח וננחש ערך כלשהוא עבור $\mathcal{V}^\pi(S_t)$ וניחוש זה יהיה גרוע. אפשר מעט לשפר את הניחוש באמצעות ניחוש $\mathcal{V}(S_{t+1})$ ושימוש ב-reward R_{t+1} שהתקבל עבור אותו מצב S_t, שבעצם מספק מידע כלשהוא על מצב זה. שערוך זה עדיף על ניחוש מוחלט כיוון שהאלמנט של הניחוש מקבל משקל נמוך יותר עקב המכפלה ב- γ, ויש יותר משקל ל-R_{t+1} שמספק מידע אמיתי על המצב S_t. צריך לשים לב שאנו מנסים לשערך את ה-Value function מתוך הערכים שלה בעצמה. מאתחלים את כל הערכים במספרים כלשהם (למשל – וקטור של 0), ואז עוברים צעד צעד ומעדכנים את הניחושים בעזרת התגמולים. אינטואיטיבית זה נראה מעט משונה, אך מסתבר שפתרון זה הוא אחד הכלים החזקים בבעיות RL. באופן פורמלי האלגוריתם נראה כך:

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Initialize $V(s)$ arbitrarily (e.g., $V(s)=0$, for all $s \in \mathcal{S}^+$)
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
    $S \leftarrow S'$
  until S is terminal

איור 11.6 אלגוריתם Temporal Difference (TD) עבור שערוך ה-Value function בהינתן Policy. מנחשים ערך עבור כל מצב ואז באופן איטרטיבי משפרים את הניחושים בעזרת שערוך התלוי ב-reward ובערך המצב הבא.

מגדירים את השגיאה של ה-TD כהפרש שבין הניחוש עבור ערך המצב לבין השערוך שלו:

$$\delta_t = R_{t+1} + \gamma \mathcal{V}(S_{t+1}) - \mathcal{V}(S_t)$$

בשונה משערוך MC, השערוך בשיטת TD הוא בעל הטיה, אך השונות קטנה יותר. בנוסף, כיוון שבכל צעד מבצעים שיפור לערך של מצב, תהליך השערוך יותר מהיר מאשר ב-MC. הרבה פעמים מוסיפים פרמטר α לשערוך (כפי שמופיע באיור 11.6):

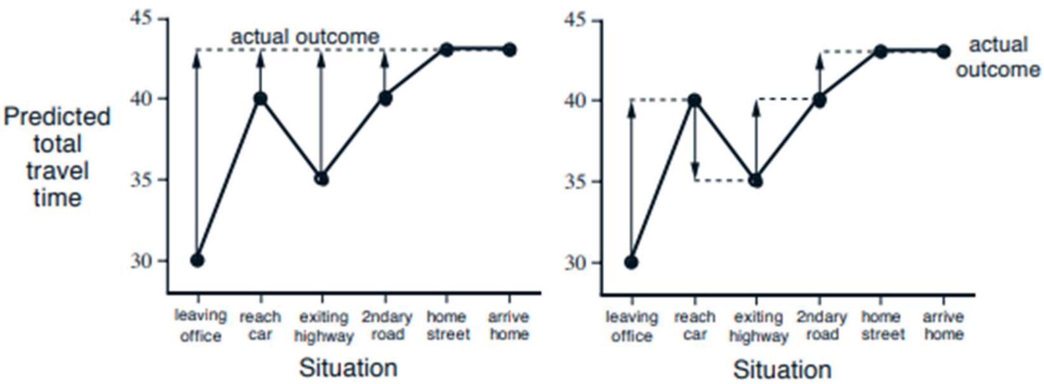
$$\mathcal{V}(S_t) = \mathcal{V}(S_t) + \alpha \cdot [R_{t+1} + \gamma \mathcal{V}(S_{t+1}) - \mathcal{V}(S_t)]$$

עבור ערכי α מתאימים, המשערך מתכנס לתוחלת האמיתית.

ניתן דוגמה שתמחיש את שיטת MC ושיטת TD ואת היחס ביניהם: נהג יצא מהמשרד שלו ונסע לביתו, ובדרך הוא ניסה לשערך את זמן ההגעה שלו וקיבל את הזמנים הבאים:

שערוך זמן הנסיעה הכולל	שערוך הזמן שנותר לנסיעה	כמה זמן עבר	מצב
30	30	0	יציאה מהמשרד
40	35	5	הליכה למכונית תחת גשם
35	15	20	הגעה לכביש מהיר
40	10	30	נסיעה מאחורי משאית
43	3	40	הגעה לרחוב של הבית
43	0	43	הגעה הביתה

נשרטט את שני המשערכים:



איור 11.7 שערוך MC (משמאל) ושערוך TD (מימין) ביחס לתצפית הנתונה.

כיוון ששיטת MC מספקת ערך לאחר ריצה שלמה, אז ניקח את זמן ההגעה בפועל של הנהג וניתן את הערך הזה לכל מצבי הביניים. שיטת TD לעומת זאת מעדכנת את הערך בכל מצב בהתאם למצב הבא. ניתן לראות שהשונות בשערוך TD קטנה מזו שהתקבלה בשערוך MC.

ניתן להסתכל על שיטת TD כבעיית רגרסיה "דינמית":

עבור כל צעד נרצה ש-$\mathcal{V}(S_t)$ יהיה שווה למשוואות בלמן, כלומר נרצה לשערך את $\mathcal{V}(S_t)$ כך שיתקיים:

$$\mathcal{V}(S_t) = \mathbb{E}_\pi[R_{t+1} + \gamma\mathcal{V}(S_{t+1})|s_t = S_t]$$

עבור בעיה זו נוכל להגדיר פונקציית מחיר (Loss) מתאימה:

$$L = \frac{1}{2}\left(\mathbb{E}_\pi[R_{t+1} + \gamma\mathcal{V}(S_{t+1})|s_t = S_t] - \mathcal{V}(S_t)\right)^2$$

את הפונקציה הזו ניתן למזער על ידי דגימות סטוכסטיות של $R_{t+1} + \gamma\mathcal{V}(S_{t+1})$. נשים לב לדבר חשוב – המטרה שלנו היא **לשערך את ההווה באמצעות העתיד ולא להיפך**, כיוון שהעתיד הוא בעל יותר מידע – הוא ראה reward ומצב חדש. הבחנה זו משפיעה על איך שאנחנו רוצים שפונקציית המחיר תתנהג – אנחנו רוצים ש-$\mathcal{V}(S_t)$ יתקרב לערך של $\mathbb{E}_\pi[R_{t+1} + \gamma\mathcal{V}(S_{t+1})|s_t = S_t]$ ולא להיפך. בדוגמה של הנהג שמשערך את זמן הנסיעה – הוא רוצה לעדכן את השערוך של כל מצב בהתאם למצב הבא. אם למשל ההערכה שלו בזמן t הינה 30 דקות ואז הוא מגיע לפקק ומעדכן את ההערכה ל-35 דקות, אז הוא ירצה לתקן את השערוך הקודם $\left(\mathcal{V}(S_t)\right)$ כך שיהיה דומה לשערוך הנוכחי $\left(R_{t+1} + \gamma\mathcal{V}(S_{t+1})\right)$, ולא לעדכן את השערוך הנוכחי כך שיהיה דומה לקודם. בכדי לדאוג לכך, נתייחס לעתיד כקבוע ולא נחשב עבורו גרדיאנט (למרות ש-$\mathcal{V}$ מופיע בו). באופן פורמלי נוכל לנסח זאת כך:

$$T(S_t) = \mathbb{E}_\pi[R_{t+1} + \gamma\mathcal{V}(S_{t+1})|s_t = S_t]$$

$$L = \frac{1}{2}\left(T(S_t) - \mathcal{V}(S_t)\right)^2$$

ואז הגרדינאנט יהיה:

$$\frac{\partial L}{\partial \mathcal{V}} = T(S_t) - \mathcal{V}(S_t)$$

בהתאם לכך, הדגימה $R_{t+1} + \gamma\mathcal{V}(S_{t+1}) - \mathcal{V}(S_t)$ הינה שערוך סטוכסטי לגרדיאנט. כיוון שכל צעד תלוי בצעד הבא, אז המטרה $T(S_t)$ משתנה בכל צעד (אמנם קיבענו אותה בכל צעד יחיד, אך היא עדיין תלויה ב-$\mathcal{V}(S_t)$). במובן הזה בעיית הרגרסיה שהגדרנו הינה "דינמית", כיוון שהמטרה משתנה בכל צעד.

11.1.2 TD(λ)

ננסה לבחון את הקשר בין שתי השיטות שראינו. שערוך TD מבצע דגימה של ריצה ואז מעדכן את הערך של כל מצב בהתאם **למצב הבא** בלבד:

$$\mathcal{V}(S_t) = \mathcal{V}(S_t) + \alpha \cdot [\boldsymbol{R_{t+1}} + \gamma\boldsymbol{\mathcal{V}(S_{t+1})} - \mathcal{V}(S_t)]$$

בגלל ההתייחסות למצב אחד בכל פעם השונות של המשערך נמוכה, אך יש הטיה.

שיטת MC לעומת זאת דוגמת ריצה ומעדכנת את הערך של כל המצב בהתאם **לכל המצבים** שבאים לאחר מכן:

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1}R_T = \sum_{k=0}^{T-t-1}\gamma^k R_{t+k+1}$$

$$\mathcal{V}(s_t) = \mathcal{V}(s_t) + \frac{1}{N(S_t)}\left(G_t - \mathcal{V}(s_t)\right)$$

משערך זה הינו חסר הטיה, אך עם זאת השונות שלו גבוהה.

נראה כיצד ניתן לחבר בין שני המשערכים ולמצוא שיטה שתהיה אופטימלית מבחינת היחס שבין ההטיה לשונות. ניתן להכליל את שני המשערכים לנוסחה כללית באופן הבא:

$$\mathcal{V}(s_t) = \mathcal{V}(s_t) + \alpha\left(G_t^{(n)} - \mathcal{V}(s_t)\right)$$

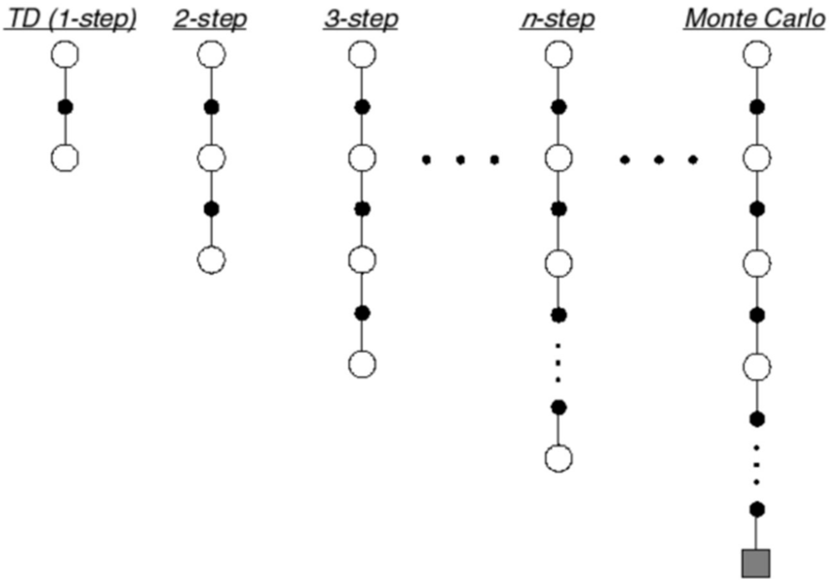
כאשר $G_t^{(n)}$ הוא הסכום של n ה-rewards הבאים:

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_n = \sum_{k=0}^{n-t-1} \gamma^k R_{t+k+1}$$

כעת נוכל להגדיר:

$$TD(n) \rightarrow \mathcal{V}(s_t) = \mathcal{V}(s_t) + \alpha\left(G_t^{(n)} - \mathcal{V}(s_t)\right)$$

ולפי סימון זה נוכל לשים לב שמשערך MC הוא למעשה $TD(n \rightarrow \infty)$ ואילו משערך TD שווה ל-$TD(1)$.



איור 11.8 משערך $TD(n)$. משערך MC הינו המקרה הפרטי עבורו $n \rightarrow \infty$ ואילו משערך TD הינו המקרה הפרטי בו $TD(n = 1)$.

אם ניקח $1 < n < \infty$ נקבל משערך "ממוצע" בין MC לבין TD. ניתן להציע גרסה יותר טובה למשערך זה תחת ההנחה שככל שמאורעות סמוכים אחד לשני כך יש להם יותר השפעה. בדומה ל-discount factor שמוריד את ההשפעה של תגמול ככל שהוא יותר רחוק מהמצב הנוכחי, כך גם כאן ניתן לכל מאורע עתידי משקל הולך וקטן. נדאג שכל המשקלים יסתכמו ל-1 ונקבל את השערוך הבא, הידוע גם בכינוי $TD(\lambda)$:

$$G_t^{\lambda} = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$$

$$\mathcal{V}(s_t) = \mathcal{V}(s_t) + \alpha\left(G_t^{\lambda} - \mathcal{V}(s_t)\right)$$

שערוך זה הוא בעל שונות גדולה יותר מאשר TD, כיוון שהוא מתחשב ביותר צעדים, אך ההטיה שלו קטנה יותר. כאמור, הוא מנסה למצע בין שני המשערכים שראינו ולאזן בין השונות להטיה.

נסכם בקצרה את מה שראינו בפרק זה. התייחסנו למקרים בהם המודל אינו ידוע ואנו רוצים לשערך אותו, והצגנו שלוש גישות המנסות לשערך את המודל בעזרת דגימות סטוכסטיות: MC, TD (שזהו מקרה פרטי של $TD(n)$) וגישה המנסה לשלב בין השתיים – $TD(\lambda)$.

לסיום נעיר ששערוך המודל כשלעצמו אינו מספיק, כיוון שהוא מספק מודל עבור הבעיה, אך הוא אינו בהכרח אופטימלי. בפרק הקודם ראינו כיצד בהינתן המודל ניתן לשפר אותו ולמצוא את האסטרטגיה האופטימלית. יכולנו לעשות זאת כיוון שידענו את המודל במלואו, מה שאיפשר לבצע בכל צעד מהלך חמדני ובכך להתכנס לבסוף לאסטרטגיה האופטימלית. במקרים בהם אנו רק משערכים את המודל, אין לנו בהכרח מידע מספיק טוב עבור כל

המצבים. מצבים ופעולות שלא נוסו כלל (או נוסו רק בפעמים נדירות), השערוך עבורם יכול להיות די גרוע, ואז השיפור באמצעות בחירה גרידית לא בהכרח יכול להביא את האסטרטגיה להיות אופטימלית. אתגר זה נקרא

11.3 Model Free Control