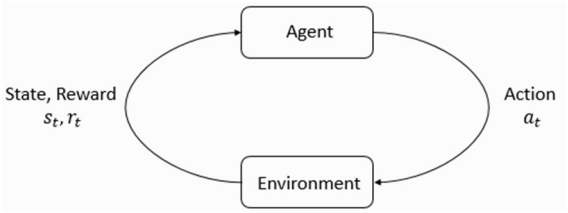
11. Reinforcement Learning (RL)

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 סט 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 דאטא 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:

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

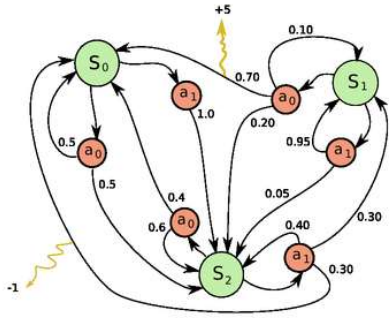
תהליך קבלת החלטות מרקובי מתואר על ידי סט הפרמטרים $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}\}$:

- 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: $T(s'|s,a) = P(s_{t+1} = s' | s_t = s, a_t = a)$.
- 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** $\sum_{t=0}^{\infty} \mathcal{R}\big(s_t, \pi(s_t)\big)$. בכדי לחשב את הסכום הזה מגדירים 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:

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

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:

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

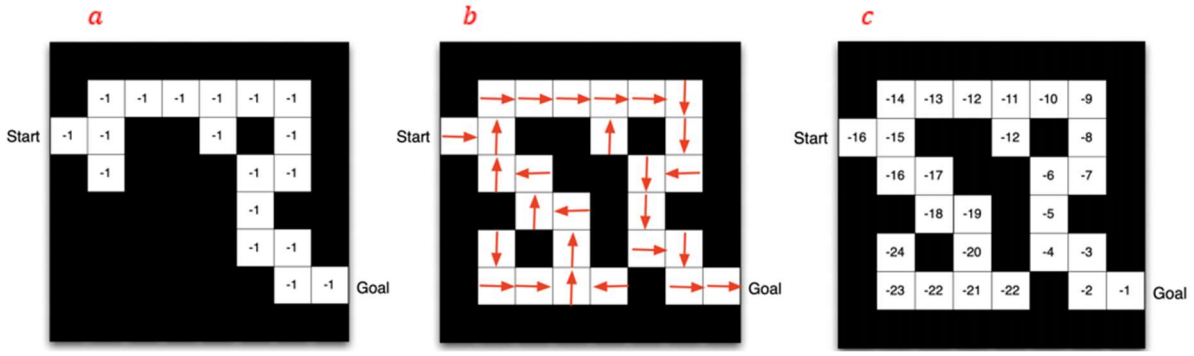
בעזרת ביטוי זה ניתן לחשב את האסטרטגיה האופטימלית, כאשר ניתן לנקוט בגישה ישירה ובגישה עקיפה. הגישה הישירה מנסה למצוא בכל מצב מה הפעולה הכי כדאית. בהתאם לכך, חישוב האסטרטגיה האופטימלית יעשה באופן הבא:

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

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:

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

ניתן לשים לב שבעוד הגישה הראשונה מתמקדת במציאת אסטרטגיה 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.

11.1.2 Planning

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:

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

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:

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

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-$Q^*(s,a)$ את הערכים של האסטרטגיה האופטימלית π^* – Optimal Value Function ו-Optimal Action-Value Function. עבור אסטרטגיה זו מתקיים:

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

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

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

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

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 – סט 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, המשתמשת בתכנון דינמי. נפתח את הביטוי של $\mathcal{V}^{\pi}(s)$ מתוך ההגדרה שלו:

$$V^{\pi}(s) = \mathbb{E}[R(\tau)] = \mathbb{E}\big[R\big(s,\pi(s)\big) + \gamma V^{\pi}(s')\big]$$

$$= \mathbb{E}\big[R\big(s,\pi(s)\big)\big] + \gamma \sum_{s'} p\big(s'\big|s,\pi(s)\big)V^{\pi}(s')$$

הביטוי המתקבל הוא מערכת משוואות ליניאריות 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:

$$V = R + \gamma TV \rightarrow V = R + \gamma TV$$

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

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, ויש שתי שיטות עיקריות לכך:

Value Iteration (VI)

חישוב $\mathcal{V}^{\pi}(s)$ מנסה לתת לכל מצב s ערך בהתאם לתוחלת התגמול שניתן להשיג משלב זה. בשלב הראשוני מאתחלים את כל המצבים s ל-0, כלומר: $\forall s, \mathcal{V}_0^*(s) = 0$, ואז מעדכנים את ערכי המצבים באופן איטרטיבי:

$$\mathcal{V}_{i+1}^*(s) = \max_a \mathbb{E}[R(s,a) + \gamma \cdot \mathcal{V}_i^*(s')] = \max_a \sum_{s' \in \mathcal{S}} \mathcal{T}(s'|s,a)[R(s,a) + \gamma \cdot \mathcal{V}_i^*(s')]$$

אגף ימין נקרא Bellman update, וקיומו מהווה תנאי מספיק למציאת $\mathcal{V}^*(s)$. ניתן להוכיח שביטוי זה מתכנס, ומספר האיטרציות הדרוש תלוי בערך של γ. לאחר מציאת $\mathcal{V}^*(s)$ ניתן לחשב את האסטרטגיה האופטימלית בעזרת כלל החלטה הנקרא policy extraction, והוא דומה לאיטרציות הקודמות:

$$\pi^*(s) = \arg\max_a \sum_{s' \in \mathcal{S}} \mathcal{T}(s'|s,a)[R(s,a) + \gamma \cdot \mathcal{V}^*(s')]$$

יש מספר חסרונות בגישה זו – הסיבוכיות יחסית יקרה ($\mathcal{O}(\mathcal{S}^2\mathcal{A})$), ערך המקסימום של הביטוי יכול להשתנות באופן תדיר, והרבה פעמים מספר האיטרציות הדרוש עד להתכנסות גדול בהרבה מהמספר הדרוש עבור מציאת אסטרטגיה אופטימלית.

Policy Iteration (PI)

גישה אחרת לחישוב האסטרטגיה האופטימלי מניחה אסטרטגיה מסוימת ועבורה מחשבת את $\mathcal{V}(s)$ עד להתכנסות של ה-values. לאחר ההתכנסות מעדכנים את האסטרטגיה בהתאם לערכי $\mathcal{V}(s)$ החדשים, ואז מבצעים את השלבים האלה שוב – מקפיאים את האסטרטגיה החדשה ומחשבים את $\mathcal{V}(s)$ ולאחר מכן שוב מעדכנים את האסטרטגיה. השלב הראשון נקרא Policy evaluation, והוא דומה ל-VI:

$$\mathcal{V}_{k+1}^{\pi_i}(s) = \sum_{s' \in \mathcal{S}} \mathcal{T}\big(s's,\pi_i(s)\big)\big[R\big(s's,\pi_i(s)\big) + \gamma \cdot \mathcal{V}_k^*(s')\big]$$

לאחר שביטוי זה מגיע להתכנסות מבצעים עדכון לאסטרטגיה:

$$\pi_{i+1}(s) = \arg\max_a \sum_{s' \in \mathcal{S}} \mathcal{T}(s'|s,a)[R(s,a) + \gamma \cdot \mathcal{V}^{\pi_i}(s')]$$

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

11.1.3 Learning Algorithms

בפרק הקודם הוסבר כיצד ניתן לחשב את האסטרטגיה האופטימלית וערך ההחזרה **בהינתן** מודל מרקובי. עיקר ההתמקדות של אלגוריתמי RL הוא 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 $\{\mathcal{S}, \mathcal{A}, \mathcal{R}\}$ ובעזרתם לחשב את $p(s_{t+1}|s_t, a_t)$ ולבחור את s_{t+1} שההסתברות שלו הכי גבוהה מבין כל האפשרויות. 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. בעוד שבגישה שאינה המשתמשת במודל יש ניסיון למצוא מקסימום לפונקציית המחיר של תוחלת ערך ההחזרה, הגישה שמבוססת מודל מנסה למצוא מסלול τ עבור פונקציית המחיר $\sum_{t=1}^{T} c(s_t, a_t)$ תהיה מינימלית.

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 לא 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 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.