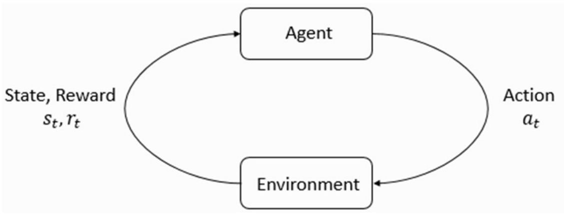
# 11. Reinforcement Learning (RL)

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 הינו 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. באופן גרפי ניתן לתאר את תהליך הלמידה כך:




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. 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, בו ההסתברות להגיע למצב מסוים תלויה רק במצב הקודם לו. תהליך זה 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-$r_t$.

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 $s_{t+1} \sim p(\cdot\,|s_t, a_t)$.

Optimal Policy – באופן כללי, עבור בעיה מסוימת נרצה למצוא את המסלול $\tau$ האופטימלי באופן ישיר או על ידי מציאת אסטרטגיה $\pi: S \to A$ הממקסמת את התגמול המצטבר $\sum_{t=0}^{\infty} \mathcal{R}\big(s_t, \pi(s_t)\big)$ (או באופן שקול – אסטרטגיה

המביאה למינימום פונקציית מחיר הבנויה ממחיר שיש עבור כל פעולה). 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 הוא קומבינציה (לינארית בדרך כלל) של תגמולים, כאשר יש מספר קומבינציות נפוצות, וההחלטה איזו קומבינציה עדיפה יכולה להיות משמעותית בהצלחת המודל:

Finite Horizon – עבור פרמטר $T < \infty$, נגדיר את ה-Return כסכום של $T$ התגמולים הראשונים:

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

Infinite discount return – נגדיר פרמטר $\gamma \in (0,1)$, ובעזרתו נגדיר את ה-Return כסכום האינסופי הבא:

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

כיוון ש-$r_t \in [0,1]$, הסכום חסום על ידי $\frac{1}{1-\gamma}$.

Average reward – גבול התוחלת המנורמל של $T$ התגמולים הראשונים:

$$Return = \lim_{T\to\infty} \frac{1}{T} \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t\right]$$

בעזרת ערך ההחזרה ניתן להגדיר את המטרה של למידה מבוססת חיזוקים. ההסתברות לקבל מסלול מסוים תחת אסטרטגיה מסוימת הינה ההסתברות של המצב הראשון כפול ההסתברות לבחור פעולה $a_1$ שתוביל למצב $s_1$ וכך הלאה עד $s_T$:

$$p(\tau|\pi) = \rho_0(s_0) \prod_{t=1}^{T-1} p(s_{t+1}|s_t, a_t)\pi(a_t|s_t)$$

כעת פונקציית המטרה תהיה התוחלת של ערך ההחזרה:

$$\mathcal{J}(\pi) \equiv \mathbb{E}[R(\tau)] = \int_{\tau} p(\tau|\pi)R(\tau)$$

אלגוריתמים מבוססי RL פועלים באחת משתי דרכים:

א. מקסום פונקציית המטרה, כלומר ניסיון להביא למקסימום את התוחלת של ערך ההחזרה:

$$\pi^* = \arg\max_{\pi} \mathcal{J}(\pi)$$

ב. חיפוש המסלול $\tau$ בעל המחיר הנמוך ביותר:

$$\min_{a_1,\dots,a_T} \sum_{t=1}^{T} c(s_t, a_t), \quad s.t\ s_t = s_{t+1} = f(s_t, a_t)$$

ניתן לשים לב שבעוד הגישה הראשונה מתמקדת במציאת אסטרטגיה אופטימלית על בסיס התוחלת של ערך ההחזרה, הגישה השנייה לא מתייחסת למדד זה אלא מנסה למצוא באופן מפורש את המודל ובעזרתו את המסלול האופטימלי.

### 11.1.2 Planning

לאחר שהגדרנו את המטרה של למידה מבוססת חיזוקים, ניתן לדבר על שיטות למציאת מסלול אופטימלי ושיטות למציאת אסטרטגיה אופטימלית. בפרק זה נתייחס למקרה הספציפי בו נתון מודל מרקובי עם כל הפרמטרים שלו, ונעשה זאת בעזרת ההגדרות הבאות:

On-Policy Value Function – התוחלת של ערך ההחזרה עבור אסטרטגיה נתונה $\pi$, 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.

On-Policy Action-Value Function – התוחלת של ערך ההחזרה עבור אסטרטגיה נתונה $\pi$, כאשר במצב $s$ מבצעים את פעולה $a$, ולאחר מכן ממשיכים לפי האסטרטגיה $\pi$:

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

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-$\mathcal{Q}^*(s,a)$ את הערכים של האסטרטגיה האופטימלית $\pi^*$ – 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 $\pi^*$ הינה האופטימלית עבור כל מצב $s$.

כעת נניח שנתון מודל מרקובי עם כל הפרמטרים שלו – סט המצבים והפעולות, הסתברויות המעבר והתגמול עבור כל פעולה, ומעוניינים למצוא אסטרטגיה אופטימלית עבור מודל זה. ישנן שלוש גישות עיקריות לפתרון בעיה זו – חישוב $\mathcal{V}^{\pi}(s)$ האופטימלי (Value iteration) וחישוב $\mathcal{Q}^{\pi}(s,a)$ האופטימלי (Policy iteration), ותכנות לינארי.

## 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 $\gamma$. 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.

Linear Programming (LP)

תכנון לינארית היא גישה המנסה למדל את הבעיה כבעיית אופטימיזציה של אוסף של אילוצים לינאריים. ישנם אלגוריתמים שונים של תכנון לינארי, וחלקם מתאימים גם לבעיות של MDP. נגדיר אינדיקטור $x_t(s,a) \in \{0,1\}$ שערכו 1 אם בזמן $t$ אנחנו נמצאים במצב $s$ ונוקטים בפעולה $a$, ואחרת ערכו 0. בנוסף נגדיר משתנה טרמינלי $x_T(s) \in \{0,1\}$ שערכו 1 אם כעת אנחנו נמצאים במצב $s$. כעת נוכל לרשום אילוצים ביחס למשתנים אלו:

$$\sum_a x_t(s,a) = \sum_{s',a',f_{t-1}(s',a')=s} x_{t-1}(s',a')$$

$$x_T(s) = \sum_{s',a',f_{T-1}(s',a')=s} x_{T-1}(s',a')$$

פונקציית המטרה לה נרצה למצוא מקסימום הינה פונקציית ההחזרה, ונרשום אותה בעזרת המשתנים שהגדרנו:

$$\sum_{t,s,a} r_t(s,a)\, x_t(s,a) + \sum_s r_T(s) x_T(s)$$

בעזרת האילוצים הקיימים ניתן למצוא אופטימום לפונקציית המטרה, וכך לחשב את האסטרטגיה האופטימלית.

### 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 $\tau$ 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 $\theta$ 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 $\theta$ 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 $\pi$.
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.