استفاده از یادگیری تقویتی برای حفظ تعادل روبات $\text{Pendubot}^{\text{TM}}$

گزارش فنی پروژه‌ی تحقیقاتی
استاد: دکتر بهزاد صمدی

محمد نخبه زعیم، محسن فلاحی، دانیال خشابی

دانشگاه صنعتی امیرکبیر(پلی تکنیک تهران)، دانشکده ی مهندسی برق، آزمایشگاه کنترل صنعتی

{nokhbeh100,mfalahi13,d.khashabi}@gmail.com

چکیده

در این گزارش فنی، خلاصه‌ای از پیشرفت پروژه‌ی حفظ تعادل روبات Pendubot ارائه خواهدشد. در ابتدا معادلات دینامیکی روبات Pendubot معرفی می‌شوند. در ادامه کلیات یادگیری تقویتی معرفی خواهدشد. سپس دو دسته از الگوریتم‌های یادگیری تقویتی بامدل و بدون‌مدل معرفی می‌شود. با توجه به ماهیت پیوسته-ی حالت‌ها در سیستم‌های واقعی، برخی از روش‌های مهم تقریب تابع بررسی خواهندشد. در واقع از چنین ساختارهایی به عنوان هسته‌ی یادگیری تقویتی استفاده خواهدشد. در انتها نیز نتایج برخی از شبیه‌سازی‌ها روی چندین مساله ارائه شده‌است.

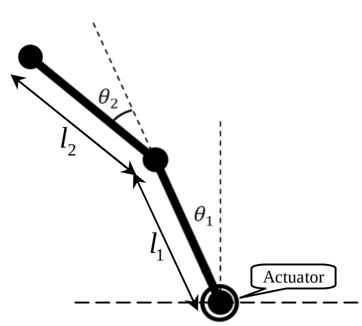
واژه‌های کلیدی: یادگیری تقویتی، حفظ تعادل، روبات Pendubot.

۱. مقدمــه

مساله‌ی حفظ تعادل از اساسی‌ترین و مهم‌ترین مسائل مهندسی کنترل است. به علت ماهیت پایداری فرار آن، رسیدن به آن، بخصوص در پیاده‌سازی واقعی بسیار دشوار است. مساله‌ی حفظ تعادل روبات Pendubot مساله‌ای چالش برانگیز است که به علت وجود معادل‌های کاربردی متعدد برای آن، بسیار مورد توجه قرار گرفته‌است. نمونه‌ای از توجهات به این مساله را می‌توان در یافت. در اینجا سعی‌شده‌است با استفاده از یادگیری تقویتی Pendubot از حالت افتاده، به حالت ایستاده انتقال یابد.

۲. تعریف دقیق صورت مساله

در این قسمت معادلات حاکم بر یک روبات Pendubot را بررسی می‌کنیم. این روبات شامل دو بازو است که به هم متصل شده‌اند. در انتهای پایین‌ترین بازو، موتوری قرار دارد که می‌تواند در هر لحظه در دو جهت حرکت کند(شکل- ۱). در حالتی که موتور به‌جای قرار گرفتن در ابتدای بازوی اول، در نقطه‌ی اتصال دهنده‌ی بازوی اول و دوم قرار گیرد، روبات مورد نظر Acrobot نامیده می‌شود[1].



شکل- ۱-تصویر مدل روبات Pendubot

بازوی اول دارای طول l_1 و مرکز جرم l_{c1} و m_1 همچنین جرم است؛ بصورت مشابه بازوی دوم دارای طول l_2 و مرکـز جـرم l_{c2} و جرم m_2 می‌باشد. می‌توان معادلات دینامیکی سیستم را به این‌صورت نوشت:

$$\begin{cases} d_{11}\ddot{\theta}_1 + d_{12}\ddot{\theta}_2 + h_1 + \phi_1 = \tau \\ d_{12}\ddot{\theta}_1 + d_{22}\ddot{\theta}_2 + h_2 + \phi_2 = 0 \end{cases}$$

اگر مقدار پارامترهای مساله بصورت ذیل باشد:

$$d_{11} = m_1 {l_{c1}}^2 + m_2\left(l_1^2 + l_{c2}^2 + 2l_1 l_{c2}\cos(\theta_2)\right) + I_1 + I_2$$

$$d_{22} = m_2 l_{c2}^2 + I_2$$

$$d_{12} = m_2\left(l_{c2}^2 + l_1 l_{c2}\cos(\theta_2)\right) + I_2$$

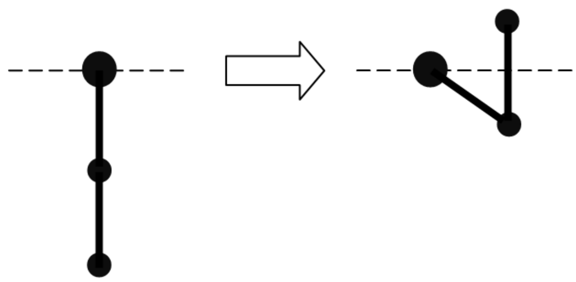
$$h_1 = -m_2 l_1 l_{c2}\sin(\theta_2)\dot{q}_2^2 - 2m_2 l_1 l_{c2}\sin(\theta_2)\dot{q}_2\dot{q}_1$$

$$h_2 = m_2 l_1 l_{c2}\sin(\theta_2)\dot{q}_1^2$$

$$\phi_1 = (m_1 l_{c1} + m_2 l_1) g\cos(\theta_1) + m_2 l_{c2} g\cos(\theta_1 + \theta_2)$$

$$\phi_2 = m_2 l_{c2}\,\mathrm{g}\cos(\theta_1 + \theta_2)$$

هدف در این مساله این است که روبات را از حالت عادی، به حالتی منتقل کنیم که بازوی دوم در حالت ایستاده قرار گیرد.



شکل- ۲-انتقال روبات از حالت طبیعی به حالت متعادل ایستاده

۳. یادگیری تقویتی

یادگیری تقویتی مدلی از یادگیری را ارائه می‌کند که در آن هدف، آموزش به سیستم از روی تکرار آزمایش اسـت. بـه ازای هـر عمل مناسب به سیستم پاداش داده شده و به ازای هر عمل مناسب‌تر، پاداش[۱] بیشتری به سیستم داده خواهدشد؛ بر عکس بـه ازای اعمال نامطلوب، مقداری پاداش کمتری به سیستم داده می‌شود. پـاداش سیسـتم در لحظـه‌ی t-ام را بـا $r_t(s,a)$ نشـان می‌دهیم که در آن s، حالت سیستم و a، عملی است که در حالت مورد نظر انجام خواهدشـد. در بسـیاری از مواقـع مـی‌تـوان پاداش را تنها تابعی از حالت سیستم s تعریف کرد. اصطلاحا به این پاداش «پاداش آنی یا لحظه‌ای[۲]» گفته می‌شود. چـرا کـه از آن برای ارزیابی شرایط موجود و بدون توجه به گذشته و آینده استفاده می‌شود. معمولا «تابع پاداش» توسـط آموزنـده تعریـف می‌شود.

با توجه به اینکه در یادگیری تقویتی اغلب اعمالی را مد نظر داریم که نتیجه‌ی آنها وابسته به بهینه بودن مجموعـه‌ای از اعمـال پشت سر هم دارد، لازم است تک‌تک پاداش‌های آنی، تا حد امکان بیشترین مقدار خود را داشته‌باشند. در نهایـت هـدف نهـایی

1 Reward
2 Immediate reward

در یک مساله‌ی یادگیری تقویتی افزایش مجموع پاداش‌های آنی است. چرا که با این کـار سـعی بـر افـزایش تـک‌تـک مقـادیر پاداش‌های لحظه‌ای داریم. متداول‌ترین تعریف برای مجموع پاداش‌های آنی مطابق رابطه‌ی (۱) است که به «مجمـوع تخفیـف-یافته‌ی پاداش[۳]» مشهور است.

$$R = \lim_{h \to \infty} E\left[\sum_{t=0}^{h} \gamma^t r_t(s,a) \right] \tag{۱}$$

در رابطه‌ی (۱) منظور از $t=0$ لحظه‌ی کنونی است. لذا تمامی $t>0$ معـرف لحظـات آینـده‌انـد. همچنـین عامـل γ ضـریب تخفیف[۴] نام دارد که عددی در محدوده‌ی $0 \le \gamma < 1$ قرار دارد. هدف از درنظرگرفتن چنین ضریبی چنـدین دلیـل مـی‌توانـد داشته باشد:

- بـا توجـه بـه اینکـه مقـادیر $r_t(s,a)$ مـی‌توانـد هـر مقـداری باشـند، هـیچ تضـمینی بـر محـدودبـودن مجمـوع $\lim_{h \to \infty} E\left[\sum_{t=0}^{h} r_t(s,a) \right]$ وجود ندارد. در صورتی که بتوان کران بالایی را برای $r_t(s,a)$ در نظر گرفت، بطوریکـه اگـر $|r_t(s,a)| \le M$ با قرار دادن ضریب تخفیف γ، همگرایی رابطه‌ی (۱) اثبات می‌شود.
- با قراردادن ضریب تخفیف γ، به اعمالی که مربوط به آینده‌ی نزدیک هستند، اهمیت بیشتری داده می‌شود.
- از نظر عملی چون پیاده‌سازی بی‌نهایت جمع امکان‌پذیر نیست با کوچک‌ترشـدن عامـل γ^t بـه ازای مقـادیر t بـزرگ، می‌توان با تقریب، تنها به محاسبه‌ی جمع پاداش چندین لحظه‌ی آینده پرداخت. برای مثال بـه ازای $\gamma = 0.9$ اعمـال محاسبات تنها روی ۲۰ لحظه‌ی آینده، کاری معقول به نظر می‌رسد.

صورت دیگری از رابطه‌ی (۱) را می‌توان به شکل رابطه‌ی (۲) نوشت. در این رابطه عامل تخفیف γ صرف نظر شده‌اسـت و بـرای تضمین همگرایی از میانگین‌گیری[۵] استفاده شده‌است.

$$R = \lim_{h \to \infty} E\left[\frac{\sum_{t=0}^{h} r_t(s,a)}{h} \right] \tag{۲}$$

هرکدام از روابط (۱) و (۲) می‌توانند مزیت‌هایی نسب به همدیگر داشته‌باشند؛ بسیاری از پدیده‌ها نیازمند رخداد اعمال متوالی زیادی هستند که نمی‌توان بین آنها تفاوت قائل شد. در واقع هر عمل اشتباه می‌تواند به خارج‌شدن سیستم از مسیرحرکت درست شود. حال آنکه این عمل می‌تواند در آینده‌ای نزدیک صورت گیرد یا در آینده‌ای دور. البته چون در عمل توانایی پیش-بینی آینده برای تصمیم‌گیرنده به صورت کامل امکان‌پذیر نیست، ممکن است پیش‌بینی‌های نادرستی از انجام عمل در آینده انجام دهیم. لذا با قراردادن ضریب تخفیف γ اهمیت بیشتری به اعمال در آینده‌ی نزدیک نسبت به آینده‌ی دور می‌دهیم. با توجه به آنچه گفته‌شد می‌توان عامل تخفیف γ را با توجه میزان توانایی خود در پیش‌بینی حالت آینده تنظیم کنیم. هرچقدر که محیط قطعی‌تر و دقیق‌تر باشد، می‌توان این ضریب را به مقداری نزدیک یک تنظیم کرد. در غیر اینصورت هرچه محیط غیرقطعی‌تر و همراه با خطا باشد، باید به آن مقادیر کمتر نسبت داد.

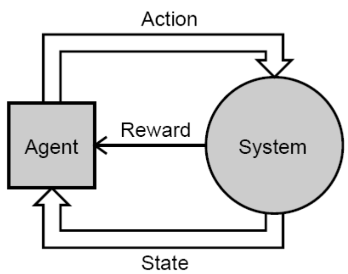
در سیستم‌هایی که عمل مورد نظر شامل زمان محدودی است، می‌توان مدل محموع پاداش‌ها را بصورت «مجموع زمان محدود[۶]» مطابق رابطه (۳) تعریف کرد که در آن h_0 مقداری مشخص و محدود است.

3 Cumulative discounted reward
4 Discount factor
5 Average cumulative reward
6 Finite horizon cumulative reward

$$R = E\left[\sum_{t=0}^{h_0} r_t(s,a)\right] \qquad (۳)$$

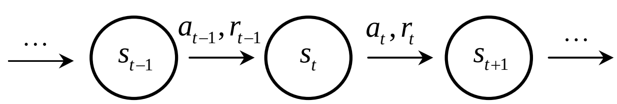
۴. فرایند تصمیم‌گیری مارکوف

با توجه به فرمول‌بندی اولیه‌ی یادگیری تقویتی که در قسمت قبل شرح داده‌شد، عمل بهینه‌سازی روی محیطی انجام می‌گیرد که در ارتباط دائم محیط و تصمیم‌گیرنده است. با تقریب بسیار مناسبی می‌توان ساختار محیط را توسط یک فرایند تصمیم‌گیری مارکوف[۷] توصیف کرد. در ادامه تعریف یک فرایند تصمیم‌گیری مارکوف یا MDP ارائه می‌شود.



شکل- ۳-مدل ارتباط عامل‌تصمیم‌گیرنده و سیستم(محیط)

یک محیط مارکوفی به محیطی گفته می‌شود که حالت آینده‌ی سیستم، تنها به حالت گذشته‌ی سیستم و عملی که در آن حالت انجام می‌شود، بستگی داشته‌باشد. بنابرین می‌توان حالت‌های سیستم را مطابق شکل- ۴ توصیف کرد. چنین پدیده‌هایی تحت عنوان «زنجیره‌ی مارکوف» توصیف و بررسی می‌شوند. چرا که با وابسته‌بودن هر حالت به حالت/عمل لحظه‌ی قبل، با دانستن حالت سیستم در زمانی دلخواه و مشخص‌بودن سیاست انتخاب عمل تصمیم‌گیرنده، حالت‌های آینده‌ی آن را می‌توان پیش‌بینی کرد.



شکل- ۴-مدل زنجیره‌ی مارکوف

یک فرایند تصمیم‌گیری مارکوف از چهارتایی $\{S, A, r, T\}$ تشکیل شده‌است. $S = [s_1, s_2, s_3, \ldots]$ برداری است که شامل تمامی حالاتی است که سیستم می‌تواند در آن قرار داشته باشد. $A = [a_1, a_2, a_3, \ldots]$ مجموعه‌ی تمامی اعمالی است که عامل می‌تواند انجام دهد. r تابع پاداش آنی است که بصورت $r: S \times A \to \mathbb{R}$ تعریف می‌شود. T تابع انتقال[۸] نام دارد که بصورت $T: S \times A \to \mathrm{P}(s)$ تعریف می‌شود. تابع انتقال حالت آینده‌ی سیستم یا s' را با توجه به حالت و عمل کنونی مشخص می‌کند. در اینجا فرض کرده‌ایم که عمل انتقال به حالت بعدی بصورت یک بردار احتمالی بصورت $\mathrm{P}(s) = [\Pr(s_1), \Pr(s_2), \Pr(s_3), \ldots]$ است که نشان می‌دهد با حضور در حالت s و انجام عمل a احتمال حضور در حالت s_i چقدر است. در حالت کلی این تابع می‌توان احتمالی[۹] یا قطعی[۱۰] باشد. همچنین هرکدام از مجموعه‌های S و A نیز می‌توانند گسسته یا پیوسته باشند.

---

7 Markov Decision Process(MDP)
8 Transition function
9 Stochastic

در نهایت پس از مدل‌سازی محیط، هدف بدست‌آوردن یک سیستم تصمیم‌گیری[11] بهینه است. یک سیاست π را به صورت $\pi : S \to A$ تعریف می‌کنیم. تابع سیاست مشخص می‌کند در هر حالت سیستم چه عملی باید انجام شود.

$$a = \pi(s) \tag{۴}$$

۵. یادگیری بامدل

با توجه به مشخص‌بودن تابع انتقال T می‌توان آنالیز مسائل را به دو دسته‌ی بامدل[12] و بی‌مدل[13] تقسیم کرد. یادگیری بامدل بدین معنی است که در محیطی عمل می‌کنیم که می‌دانیم با قرارداشتن در حالت s و انجام عمل a با احتمال مشخصی به حالت s' در آینده خواهیم‌رفت. تمامی روش‌هایی از آموزش که در این قسمت معرفی می‌شوند، «بامدل» هستند.

۵.۱. تابع ارزش[14]

با توجه به رابطه‌ی (۲) اگر این رابطه را میزان ارزش[15] تصمیم‌گیری تحت سیاست π بنامیم، تابع ارزش را به اینصورت از روی رابطه‌ی (۲) تعریف می‌کنیم:

$$V^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t r_t(s, \pi(s))\right] \tag{۵}$$

با توجه به آنچه که گفته‌شد هدف ماکزیمم‌کردن تابع ارزش $V(s)$ می‌باشد؛ در واقع به دنبال سیاستی بهینه هستیم که باعث شود مقدار (۵) ماکزیمم گردد. اگر سیاست بهینه را با π^* نشان می‌دهیم و تابع ارزشی را که تحت سیاست بهینه انجام شود را با $V^*(s) = V(s)\big|_{\pi=\pi^*}$ نشان دهیم داریم:

$$V^*(s) = \max_{\pi=\pi^*} E\left[\sum_{t=0}^{\infty} \gamma^t r_t\left(s, \pi(s)\right)\right] \tag{۶}$$

در اینصورت سیاست بهینه برابر خواهدبود با:

$$\pi^*(s) = \arg\max_{\pi} V^{\pi}(s) = \arg\max_{\pi} E\left[\sum_{t=0}^{\infty} \gamma^t r_t\left(s, \pi(s)\right)\right] \tag{۷}$$

در صورتی که رابطه‌ی (۵) را ساده کنیم به رابطه‌ی زیر خواهیم‌رسید که ارتباط بین ارزش دو حالت متوالی را برقرار می‌سازد. جزئیات اثبات این رابطه در [2] آمده‌است.

$$V^{\pi}(s) = r\left(s, \pi(s)\right) + \gamma \sum_{s' \in S} V^{\pi}(s') T(s, \pi(s), s') \tag{۸}$$

لذا تابع ارزش متناسب با سیاست بهینه به صورت زیر تبدیل خواهدشد:

$$V^*(s) = \max_{a}\left[r(s,a) + \gamma \sum_{s' \in S} V(s') T(s,a,s')\right] \tag{۹}$$

و تعریف سیاست بهینه نیز به‌صورت زیر تغییر می‌کند:

$$\pi^*(s) = \arg\max_{a}\left[r(s,a) + \gamma \sum_{s' \in S} V(s') T(s,a,s')\right] \tag{۱۰}$$

---

10 Deterministic
11 Decision policy
12 Model-based
13 Model-free
14 Value function
15 Value

معادلات (۹) و (۱۰) به معادلات بهینگی بلمن[16] معروف هستند. نکته‌ای که در معادلات بهینگی اخیر نهفته است، این است که با شروع روی زنجیره‌ی اعمال بهینه و انجام یک عمل، ادامه‌ی زنجیره با فرض آغاز حرکت از آن نقطه، در ادامه‌ی زنجیره‌ی قبلی خواهدبود. در واقع با این فرض است که می‌توان گفت در صورت تشخیص زنجیره‌ای به عنوان مجموعه‌ی اعمال برای رسیدن به بهینه‌ترین حالت، اگر تابع انتقال در لحظه‌ی آینده تغییر نکند، در لحظه‌ی آینده همان زنجیره به عنوان زنجیره‌ی اعمال بهینه انتخاب خواهدشد.

در صورتی که فرض کنیم تابع انتقال بصورت قطعی است، می‌توان معادلات بهینگی بلمن را ساده‌تر نوشت. در این حالت می-توان معادله‌ی انتقال را بصورت رابطه‌ی نوشت. به عبارت دیگر با مشخص‌بودن حالت کنونی s و عمل مورد نظر a برای انجام، حالت آینده s' به صورتی قطعی مشخص خواهدبود.

$$s' = T(s,a) \tag{۱۱}$$

در این حالت معادلات (۹) و (۱۰) به‌ترتیب به صورت (۱۲) و (۱۳) ساده خواهندشد.

$$V^*(s) = \max_a \left[r(s,a) + \gamma V(s') \right] \tag{۱۲}$$

$$\pi^*(s) = \arg\max_a \left[r(s,a) + \gamma V(s') \right] \tag{۱۳}$$

۵.۲. تابع Q [17]

تابع Q را مشابه تابع ارزش تعریف می‌کنیم، با این تفاوت که تابع Q علاوه بر اینکه تابع حالت سیستم s است، تابع عمل a نیز هست. رابطه‌ی بین تابع ارزش و تابع Q به اینصورت تعریف می‌شود:

$$V^\pi(s) = Q(s, \pi(s)) \tag{۱۴}$$

$$Q^\pi(s,a) = r(s,a) + \gamma \sum_{s' \in S} T(s,a,s') Q^*(s', \pi(s')) \tag{۱۵}$$

لذا برای استراتژی بهینه داریم:

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{a'} Q^*(s',a') \tag{۱۶}$$

$$\pi^*(s) = \arg\max_a Q^*(s,a) = \arg\max_a \left[r(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{s'} Q^*(s',a') \right] \tag{۱۷}$$

با درنظرگرفتن تابع ارزش و تابع Q برای استراتژی بهینه داریم:

$$V^*(s) = \max_a Q^*(s,a) \tag{۱۸}$$

در صورتی که فرض کنیم تابع انتقال بصورت قطعی است، می‌توان معادلات بهینگی برای تابع Q را ساده‌تر نوشت. در این حالت می‌توان معادله‌ی انتقال را بصورت رابطه‌ی نوشت. به عبارت دیگر با مشخص‌بودن حالت کنونی s و عمل مورد نظر a برای انجام، حالت آینده s' به صورتی قطعی مشخص خواهدبود.

$$s' = T(s,a) \tag{۱۹}$$

در این حالت معادلات (۱۶) و (۱۷) به‌ترتیب به صورت (۲۰) و (۲۱) ساده خواهندشد.

$$Q^*(s,a) = r(s,a) + \gamma \max_{a'} Q^*(s',a') \tag{۲۰}$$

16 Bellman optimality equations
17 Q-function

$$\pi^*(s) = \arg\max_a \left[r(s,a) + \gamma \max_{a'} Q^*(s',a') \right] \tag{۲۱}$$

۵.۳. راه‌حل صریح برای یادگیری با مدل

در صورتی که فضای حالت‌ها گسسته باشند و همچنین مدل سیستم مشخص باشد، می‌توان میزان تابع ارزش را بصورت صریح حل کرد و جواب آن را بدست آورد. با توجه به رابطه‌ی(۸) داریم:

$$V^\pi = R^\pi + \gamma T^\pi V^\pi \tag{۲۲}$$

در رابطه‌ی اخیر V^π بردار ارزش به ازاش هر حالت است و T^πبردار انتقال حالت‌ها تحت سیاست πاست. همچنین R^π بردار پاداش به ازای حالت‌های سیستم است. با حل معادله‌ی ماتریسی (۲۲) می‌توان مقدار ماتریس ارزش برای سیاست πرا بدست آورد.

$$V^\pi (1-\gamma T^\pi) = R^\pi \tag{۲۳}$$

$$V^\pi = R^\pi (1-\gamma T^\pi)^{-1} \tag{۲۴}$$

با محدود بودن تعداد سیاست‌ها، می‌توان سیاست بهینه را از حل ماتریس ارزش آنها بدست آورد.

۵.۴. راه‌حل تکراری: تکرار روی تابع ارزش[۱۸]

رابطه‌ی(۹) را در نظر می‌گیریم. می‌توان با تکرار بی‌نهایت نسبت‌دهی زیر، مقدار $V^*(s)$ را بدست آورد.

$$\forall s \in S : V_{k+1}^*(s) := \max_a \left[r(s,a) + \gamma \sum_{s' \in S} V_k(s') T(s,a,s') \right] \tag{۲۵}$$

$$\forall (s,a) \in (S,A) : Q_{k+1}^*(s,a) := r(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{a'} Q_k^*(s',a') \tag{۲۶}$$

اثبات همگرایی این تکرار در [2] آمده‌است. می‌توان الگوریتم روش تکرار روی تابع ارزش را بصورت ذیل بیان کرد[3]:

initialize $V_0(s)$ **arbitrarily**
loop until policy good enough{
  loop for $s \in S$ **{**
    loop for $a \in A$ **{**

$$Q(s,a) := r(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{a'} Q(s',a')$$

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

با محاسبه‌ی $Q^*(s,a)$ یا $V^*(s)$می‌توان استراتژی بهینه π^*را بدست آورد.

۵.۵. راه‌حل تکراری: تکرار روی سیاست[۱۹]

با داشتن هر سیاست، می‌توان مقدار تابع تابع $Q(s,a)$و تابع $V(s)$را محاسبه کرد. لذا می‌توان جستجو را روی سیاست‌ها انجام‌داد. الگوریتم تکرار روی سیاست به اینصورت است[3]:

choose an arbitrary π_0
loop until policy good enough{

18 Value iteration
19 Policy iteration

compute value function of policy π_k **:**

solve the linear equations o

$$V^{\pi_k}(s) := \max_a \left[r(s,a) + \gamma \sum_{s' \in S} V^{\pi_k}(s') T(s,a,s') \right]$$

$$\pi_{k+1}(s) := \arg\max_a \left[r(s,a) + \gamma \sum_{s' \in S} V^{\pi_k}(s') T(s,a,s') \right]$$

$$k \leftarrow k+1$$

}

۶. یادگیری مستقل از مدل

در تمامی مدل‌هایی از یادگیری تقویتی که تاکنون معرفی‌شد، فرض بر این بود که تابع انتقال حالت T مشخص‌است. در ادامه روش‌هایی از یادگیری تقویتی را مرور می‌کنیم که در آنها، یادگیری مستقل از مدل سیستم است. با توجه به اینکه در واقعیت سیستم‌ها به قدری پیچیده‌اند که امکان مدل‌سازی و تعیین تابع انتقال حالت T امکان‌پذیر نیست، لذا استفاده از مدل‌هایی که در اینجا مرور می‌کنیم، بسیار می‌توانند سودمند باشند.

۶.۱. یادگیری اختلاف محیطی[۲۰]

در این مدل روشی برای یادگیری تابع V ارائه می‌شود که نیازی به دانستن مدل محیط نیست. اگر روابط (۱۶) و (۹) را در نظر بگیریم، برای اعمال این روابط باید حتما مدل سیستم را داشته باشیم؛ چرا که نیاز به داشتن تابع انتقال $T(s,a,s')$ است. در صورتی که بتوان روشی مستقل از تابع انتقال $T(s,a,s')$ برای یاددهی به سیستم یافت، می‌توان به روش یادگیری مستقل از مدل دست یافت.

مدل یادگیری اختلاف محیطی یا $TD(0)$ روشی برای اعمال بروزرسانی عددی روی تابع V است که به صورت ذیل است:

$$V_{new}(s) := V_{old}(s) + \alpha \left(r(s,a) + \gamma V_{old}(s') - V_{old}(s) \right) \quad (۲۷)$$

می‌توان بروزرسانی فوق را به صورت دیگری نیز نوشت. مشاهده می‌شود این بروزرسانی در واقع یک میانگین‌گیری از مقادیر ارزش‌ها روی تابع V است.

$$V_{new}(s) := (1-\alpha) V_{old}(s) + \alpha \left(r(s,a) + \gamma V_{old}(s') \right) \quad (۲۸)$$

۶.۱.۱. مسیرهای شایستگی[۲۱]

این حالت حالتی تعمیم‌یافته از الگوریتم $TD(0)$ معرفی شده در قسمت قبل است که با نماد $TD(\lambda)$ نشان داده می‌شود.
تصور کنیم که حالت s_4 قرار داریم و در این حالت پاداش r_4 بسیار بزرگ را دریافت کرده‌ایم که بسیار بزرگ‌تر از اختلاف محلی $\gamma V(s_5) - V(s_4)$ است. لذا مقدار $V(s_4)$ توسط رابطه‌ی(۲۷) بسیار افزایش خواهدیافت. در این حالت شاید لازم بوده باشد که $V(s_3)$ نیز افزایش یابد. چرا که در لحظه‌ی قبل توسط رابطه‌ی $r_3 + \gamma V(s_4) - V(s_3)$ بروز شده‌است و می‌دانیم $V(s_4)$ که در آن زمان کم‌تر بوده است. به همین‌صورت باید مقدار V تمامی لحظات قبل افزایش یابند. مسیرهای شایستگی در واقع به عنوان حافظه‌ای عمل می‌کنند که با اعمال آنها، بروزرسانی تابع ارزش را تسهیل می‌بخشند. در حقیقت اگر تصمیم‌گیرنده در مسیر حرکت خود روی حالت‌ها، پاداش‌های r_{t-2} و r_{t-1} و r_t را دریافت کند، باید بروزرسانی $V(s_{t-2})$ زیر را روی تابع ارزش انجام داد:

$$V_{new}(s_{t-2}) := V_{old}(s_{t-2}) + \alpha \left[r_{t-2} + \gamma r_{t-1} + \gamma^2 r_t + \gamma^3 V_{old}(s_{t+1}) - V_{old}(s_{t-2}) \right] \quad (۲۹)$$

در حالت کلی می‌توان ساختار بروزرسانی با مسیرهای شایستگی را به صورت زیر معرفی کرد:

20 Temporal difference learning
21 Eligibility traces

$$V_{new}(s) := V_{old}(s) + \alpha\left(r(s,a) + \gamma V_{old}(s') - V_{old}(s)\right)e(s) \qquad (۳۰)$$

که در آن $e(s)$ ماتریس مسیر شایستگی است. بصورت صریح می‌توان ماتریس مسیر شایستگی را بصورت زیر تعریف کرد:

$$e(s) = \sum_{k=1}^{t} (\lambda\gamma)^{t-k} \delta_{s,s_k} \qquad (۳۱)$$

که در آن λ ضریب فراموشی است و δ_{s,s_k} تابع دلتای کرانکر است. می‌توان در هر تکرار این ماتریس را به این‌صورت بروز رسانی کرد. در واقع با حضور در هر حالت، درایه‌ی متناظر با آن حالت، حضور در آن حالت را به خاطر می‌سپارد.

$$e(s) = \begin{cases} \lambda\gamma e(s) + 1 & s = current - state \\ \lambda\gamma e(s) & else \end{cases} \qquad (۳۲)$$

چنین الگوریتمی $TD(\lambda)$ نام دارد. در حالتی که $\lambda = 0$ الگوریتم به تبدیل $TD(0)$ خواهدشد و در حالتی کـه $\lambda = 1$ الگـوریتم تمامی حالاتی را که از آن گذشته است را در نظر خواهد گرفت. به چنین متدی، «مونـت کـارلو[۲۲]» گوینـد. اگرچـه بـه ازای λ نزدیک یک، هزینه‌ی محاسباتی این الگوریتم بسیار بالاست، ولی نشان داده شده‌است که در این حالـت همگرایـی الگـوریتم بـا سرعت بسیار بیشتری انجام می‌گیرد. لذا استفاده از مسیرهای شایستگی می‌تواند عملیات آموزش را بسیار سرعت ببخشد.

۶.۲. یادگیری- Q [۲۳]

یادگیری Q از بروزرسانی زیر برای برای تابع Q استفاده می‌کند. [4]

$$Q_{new}(s,a) := Q_{old}(s,a) + \alpha\left(r(s,a) + \gamma \max_{a'} Q_{old}(s',a') - Q_{old}(s,a)\right) \qquad (۳۳)$$

در این تعریف $\alpha \in [0,1]$ ضریب یادگیری است. قاعده‌ی یادگیری Q با احتمال یک همگرا خواهدشد. بصورت معادل مـی‌تـوان قاعده‌ی یادگیری فوق را بصورت زیر نوشت. مشاهده می‌شود می‌توان رابطه‌ی زیر را به عنوان یـک میـانگین دو مقـدار در نظـر گرفت.

$$Q_{new}(s,a) := (1-\alpha)Q_{old}(s,a) + \alpha\left(r(s,a) + \gamma \max_{a'} Q_{old}(s',a')\right) \qquad (۳۴)$$

اثبات همگرایی یادگیری Q در [5] آمده‌است. لازم به تذکر است که قاعده‌ی یـادگیری تکـراری بـرای تـابع Q معـادل یـادگیری $TD(0)$ برای تابع V است.

۷. گزینه‌های مختلف برای پیاده‌سازی تقریب‌زننده‌ی درونی یادگیرنده

با توجه به اینکه در اغلب سیستم‌های واقعی فضای‌حالت سیستم و فضای اعمال تصمیم‌گیرنده پیوسته‌انـد، بایـد چـاره‌ای بـرای پیاده‌سازی الگوریتم‌ها روی آنها اندیشید. در واقع هدف طراحی ساختاری برای ایجاد و تقریـب $V(s)$ یـا $Q(s,a)$ اسـت. اولـین راهی که در استفاده از الگوریتم یادگیری تقویتی روی حالت/عمل پیوسته به ذهن می‌رسد، گسسته‌سازی روی بـازه‌ی حالـت و عمل است. می‌توان در نقاطی که دقت بیشتری مورد نیاز است، بازه‌ی گسسته‌سازی را کوچکتر در نظر گرفـت. روش گسســته-سازی چندین اشکال دارد:

- انجام گسسته‌سازی باعث از بین رفتن دقت مورد نیاز می‌شود و باعث پله‌پله‌شدن انجام اعمال می‌شود. لذا تصـمیمات و رفتارهای روبات بصورت نرم انجام نمی‌گیرد.
- شاید به نظر برسد که می‌توان مشکل اخیر را با افزایش تعداد گسسته‌سازی‌ها جبران کرد. بـا توجـه بـه اینکـه تعـداد حالت‌ها، با ضرب‌تعداد گسسته‌سازی روی هر بعد بدست می‌آید، تعداد حالت‌ها، بـا افـزایش تعـداد گسســته‌سـازی‌هـا

22 Mont Carlo methods
23 Q-learning

بصورت نمایی افزایش می‌یابد. در واقع گسسته‌سازی زیاد، باعث ایجاد افزایش بی‌رویه‌ی ابعاد[24] حالت می‌شود که باعث کندشدن الگوریتم خواهدشد. همچنین برای آموزش چنین سیستمی, به تعداد آزمایش بسیار بیشتری هم نیاز است.

لذا برای رفع مشکلات ذکرشده لازم است از ساختارهایی برای تقریب تابع یا استفاده شود. با توجه ساختار یادگیری تقویتی در صورتی که دارای ویژگی‌های زیر باشد، مناسب‌تر است:

- قابلیت آموزش به ازای تعداد نمونه‌های آموزشی کم.
- قابلیت توسعه‌ی نتایج به حالاتی که تابحال مشاهده نشده‌اند با دقت مناسب.
- قابلیت آموزش در سرعتی قابل قبول
- قابلیت پاسخ سریع در زمان نیاز؛ در صورتی که تخمین زننده کند باشد و با تاخیر پاسخ دهد، نمی‌تواند گزینه‌ی مناسبی باشد؛ تصمیم‌گیری با وقفه می‌تواند موجب از بین رفتن تعادل سیستم شود.

با توجه به ویژگی‌های ذکرشده، به چندین تقریب‌زننده که یا استفاده شده‌اند یا قصد استفاده از آن‌ها وجود دارد، اشاره می‌شود.

۷.۱. استفاده از شبکه‌ی عصبی MLP

شبکه‌ی عصبی نمونه‌ی یک تقریب‌زننده‌ی نسبتا مناسب است. اولین بار در [6] برای آموزش یک بازی تخته نرد از شبکه‌ی عصبی و یادگیری تقویتی استفاده شده‌است. از این نمونه به عنوان ماهرترین بازی تخته در دنیا نام برده می‌شود!

برخی مشکلات موجود در آموزش شبکه‌ی عصبی باعث دشواری آموزش آن می‌شوند. کندبودن سرعت آموزش Back-Propagation باعث سخت‌شدن آموزش آن می‌شود. در این شرایط می‌توان الگوریتم‌های پیشرفته‌تری برای آموزش شبکه‌ی عصبی استفاده کرد؛ از جمله Rprop که در [7] معرفی و در [8] نیز استفاده شده‌است. این الگوریتم داده‌ها را به‌صورت گروهی[25] آموزش می‌دهد.

نمونه‌هایی از پیاده‌سازی شبکه عصبی در [9] و [10] و [11] نیز ارائه‌شده‌است. استفاده از شبکه‌ی عصبی برای تقریب‌زننده‌ی تابع ارزش/Q در یادگیری تقویتی بسیار دشوار است؛ بخصوص اگر آموزش مورد نظر «بدون مدل» باشد! مطالعه‌ی روش‌ها و ترفندهای عمگرایی شبکه‌ی عصبی در این کاربرد می‌تواند بسیار مفید باشد.

۷.۲. استفاده از شبکه‌ی عصبی RBF

ویژگی تقریب محلی در استفاده از شبکه‌هایRBF بسیار مورد توجه است. می‌توان شبکه‌های عصبی برای تقریب $V(s)$ یا $Q(s,a)$ استفاده کرد. نمونه‌هایی از این پیاده‌سازی در [12] ارائه‌شده‌است.

۷.۳. استفاده از تقریب فازی

نمونه‌هایی از این پیاده‌سازی در [13] ارائه‌شده‌است.

۷.۱. استفاده از تقریب زننده‌ی CMAC

نمونه‌هایی از این پیاده‌سازی در [14] ارائه‌شده‌است.

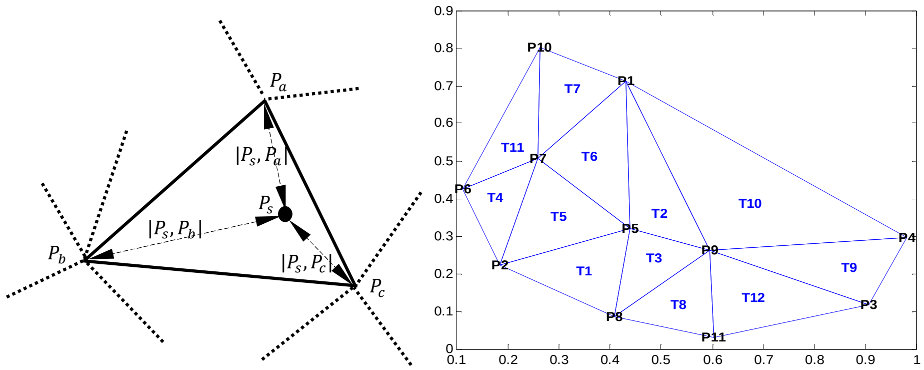
۷.۲. استفاده از تقریب زننده بر اساس مش‌بندی n-بعدی دلانی[26]

مثلث‌بندی دلانی[27]، روشی در هندسه‌ی محاسباتی برای ایجاد مثلث‌بندی تعدادی نقاط است. نمونه‌ای از این استراتژی مثلث-بندی در شکل- ۵-(الف) نشان‌داده شده‌است.

---

24 Curse of dimensionality
25 Batch
26 Delaunay n-dimensional approximation
27 Delaunay triangulation

با نسبت‌دادن مقادیر به گره‌ها و انجام درون‌یابی خطی درون هر مثلث می‌توان، تقریبی از مقدار سه راس را بدست آورد. بدین ترتیب یک تقریب‌زننده‌ی دو متغیره با دقت بالا بدست می‌آید.



(ب) (الف)

شکل- ۵- (الف)نمونه‌ای از مثلث‌بندی دالانی (ب)درون‌یابی خطی با استفاده از مثلث‌بندی دالانی

در حالت n-بعدی شکل مورد نظر تبدیل به یک سیمپلکس[۲۸] خواهدشد. در این حالت پیچیدگی زمانی الگوریتم خانه‌بندی دلانی $O(dn^2)$ است که در آن d تعداد بعد و n تعداد گره‌هاست. این تقریب‌زننده تابحال چندان مورد توجه قرار نگرفته‌است به نظر می‌رسد با بهبود آن، بتواند با سایر تقریب‌زننده‌هایی همچون شبکه‌عصبی رقابت کند. تابحال چنین تقریب‌زننده‌ای برای یک سیستم یادگیری تقویتی استفاده نشده‌است.

۸. سایر مسائل مطرح در بهینه‌سازی الگوریتم یادگیری تقویتی

۸.۱. چالش تعادل بین جستجو در فضا و انجام بهترین عمل

یکی از مهم‌ترین چالش‌ها در انجام اعمال برای یادگیری، استراتژی انتخاب اعمال برای آزمایش است. در حالت کلی دو دسته عمل می‌توان انجام داد:

- انجام بهترین عمل[۲۹] بر اساس شناختی که تابحال از محیط داشته‌ایم.
- انجام عملی غیر از عملی که به عنوان عمل بهینه می‌شناسیم. در واقع انجام چنین عملی برای کسب تجربه و جستجو[۳۰] در فضای اعمال و حالت‌ها صورت می‌گیرد.

در صورتی که توازن بین دو دسته از اعمال فوق در فاز یادگیری به درستی صورت نگیرد، یادگیری می‌تواند با مشکلات چالش برانگیزی روبرو شود. برای مثال اگر تصمیم‌گیرنده با جستجو در ناحیه‌ای محدود به این نتیجه برسد که عمل خاصی بهینه است و با تکرار آن عمل بهینه از جستجوی سایر نقاط فضای جستجو صرف نظر کند، بسیار محتمل است که از عمل بهینه‌ی دیگری صرف نظر کند.

در ادامه چندین روش برای ایجاد توازن بین انتخاب تصادفی عمل و انتخاب عمل بهینه پیشنهاد شده‌است.

۸.۱.۱. استراتژی تصمیم‌گیری ε -حریصانه[۳۱]

در این روش اگر فرض کنیم ε احتمالی نسبتا کوچک باشد، انتخاب با احتمالات زیر انجام خواهدگرفت:

---

28 Simplex
29 Exploit
30 Explore
31 ε -greedy exploration

$$\begin{cases} \Pr\left(\pi^*(s)\right) = 1-\varepsilon \\ \Pr\left(random-action\right) = \varepsilon \end{cases} \quad (۳۵)$$

می‌توان مقدار εرا به تدریج با گذر زمان کاهش داد.

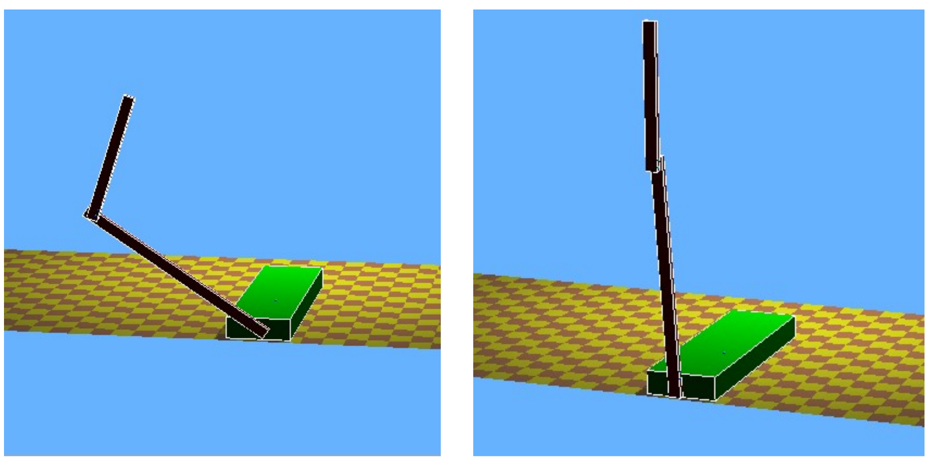
۸.۱.۲.استراتژی تصمیم‌گیری بولتزمن[۳۲]

در این روش تصمیم‌گیری به هر عمل احتمالی مطابق رابطه‌ی زیر نسبت داده می‌شود و مطابق با احتمال آن، عمل مورد نظر انتخاب می‌شود. در این رابطه $ER(a)$میزان مجموع پاداش انتظاری برای عمل مورد نظر است. مقدار T مشهور به «پارامتر دما» است که به تدریج کاهش می‌یابد. لذا به تدریج احتمال انجام بهترین اعمال بیشترین می‌شود.

$$P(a) = \frac{e^{ER(a)/T}}{\sum_{a_i \in A} e^{ER(a_i)/T}} \quad (۳۶)$$

۹. نتایج پیاده‌سازی های شبیه‌سازی

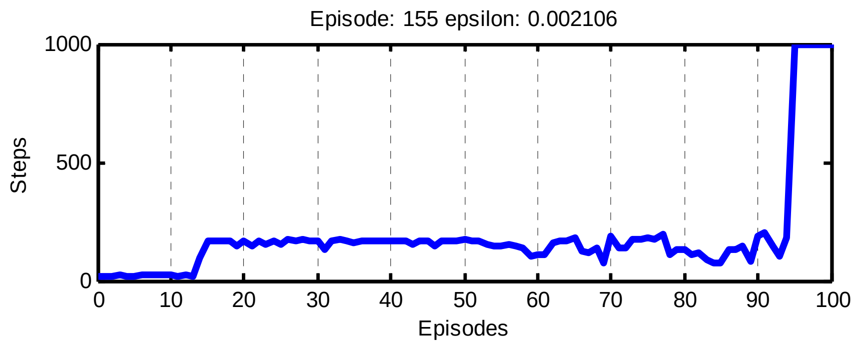
شبیه‌سازی الگوریتم یادگیری تقویتی توسط نرم‌افزار Webots صورت می‌گیرد. این نرم‌افزار امکان این را فراهم می‌کند که بتوان از مفسر MATLAB برای کدنویسی و اجرای الگوریتم استفاده کرد. لذا تمامی‌کدها در محیط MATLAB پیاده‌سازی شده‌اند.



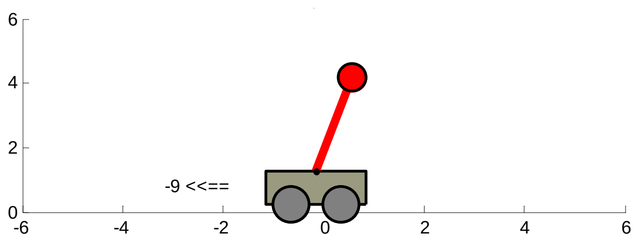
شکل- ۶-نمایش شبیه‌سازی Pendubot در محیط شبیه‌سازی Webots

در شکل- ۷ نتیجه پیاده سازی الگوریتم یادگیری Q روی مساله‌ی Cart-Pole نشان داده شده‌است. مشاهده می‌شود که توانسته‌ایم الگوریتم را پس از حدود ۹۵ بار آزمایش همگرا کنیم. باید تاکید کرد که این الگوریتم یادگیری را بدون مدل و بدون در دسترس داشتن مقادیر انجام می‌دهد. تابع مورد استفاده برای این تقریب، جدولی گسسته از مقادیر است که بروزرسانی روی آن، در هر همسایگی انجام می‌شود. شرح دقیق این روش در گزارش بعد ارائه خواهدشد.

32 Boltzmann exploration



شکل- ۷-نمودار همگرایی برای حفظ تعادل روبات Cart-Pole



شکل- ۸-مساله‌ی Cart-Pole

۱۰. پیاده‌سازی عملی

نمونه‌ی عملی ساخته‌شده از روبات Pendubot در شکل- ۹ نشان داده شده‌است. در ادامه‌ی پروژه روبات Pendubot ساخته‌شده توسط واسط به رایانه متصل خواهدشد و شبیه‌سازی‌ها بصورت عملی روی روبات مورد نظر پیاده‌سازی می‌شوند. پیش‌بینی می-شود به علت ماهیت نویزی محیط و پاسخ غیر خطی موتور، الگوریتم‌های بدون مدل نتیجه‌ی بهتری داشته‌باشند. نتـایج پیـاده-سازی در محیط واقعی در گزارش بعد ارائه خواهدشد.


شکل- ۹-نمونه‌ی واقعی روبات Pendubot ساخته‌شده

۱۱. نتیجه‌گیری و کار آینده

در این گزارش فنی، خلاصه‌ای از پیشرفت پروژه‌ی حفظ تعادل روبات Pendubot ارائه شد. همچنین مروی بر اصلی‌ترین روش-های یادگیری تقویتی شد.

کار آینده در این پروژه به تکمیل سخت‌افزار و پیاده‌سازی الگوریتم‌های مطرح‌شده اختصاص دارد. در واقع هدف این است که کارایی الگوریتم‌های مذکور در محیط شبیه‌سازی و پیاده‌سازی عملی سنجیده شود. بررسی چالش‌ها و تفاوت‌های موجود بین شبیه‌سازی و پیاده‌سازی واقعی می‌تواند منجرو به تولید روش‌ها و الگوریتم‌های جدید شود. همچنین سعی خواهدشد در روش-های پیشنهادشده بهبود ایجاد شده و سرعت همگرایی آن‌ها افزایش یابند.

با توجه به اینکه مساله‌ی حفظ تعادل روبات Pendubot مساله‌ای نسبتا دشوار است، با پیاده‌سازی موفقیت امیز الگوریتم روی آن، می‌توان راه را بر سایر مسائل مطرح در مهندسی از جمله مهندسی کنترل گشود.

۱۲. تقدیر و تشکر

نویسندگان، از مهندس مجتبی خلیجی به خاطر راهنمایی‌ها در زمینه‌ی الگوریتم یادگیری تقویتی قدردانی می‌کنند. همچنین امکانات فراهم‌شده در آزمایشگاه کنترل صنعتی دانشکده‌ی دانشگاه صنعتی امیرکبیر و حمایت جناب دکتر صمدی، پشتوانه‌ی این پروژه بوده‌اند.

۱۳. مراجع

[1] Gary Boone, "Minimum-Time Control of Acrobot," in *International Conference on Robotics and Automation* , 1997, pp. 3281--3287.

[2] Andrew G. Barto Richard S. Sutton, *Reinforcement Learning: An Introduction*.: MIT Press, Cambridge, MA, 1998.

[3] M.L.Littman, A.Moore L.P.Kaelbling, "Reinforcement Learning: A Surey," *Journal of Artificial Intelligence Research* , 1996.

[4] C. J. C. H Watkins, "Learning from delayed rewards.," Cambridge University, Cambridge, England, Doctoral thesis 1989.

[5] P Dayan C. J. C. H. Watkins, "Technical note: Q-learning," *Machine Learning*, vol. 8(3/4), pp. 279-292, 1992.

[6] Gerald Tesauro, "Temporal difference learning and TD-Gammon," *Communications of the ACM*, vol. 38, no. 3, 1995.

[7] Heinrich Braun Martin Riedmiller, "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm," in *IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS*, 1993, pp. 586--591.

[8] Martin Riedmiller, "Neural Fitted Q Iteration - First Experiences with a Data Efficient Neural Reinforcement Learning Method ," in *In 16th European Conference on Machine Learning(EMCL)*, 2005.

[9] Claude Touzet, "Neural reinforcement learning for behaviour synthesis," *Robotics and Autonomous Systems*, vol. 22, pp. 251--281, 1997.

[10] Siwei Luo, Qingyong Li Xianhua Zeng, "An associative sparse coding neural network and applications," vol. 73, no. 4-6, 2010.

[11] Shivaram Kalyanakrishnan and Peter Stone, "Batch Reinforcement Learning in a Complex Domain," in *The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*.: ACM, pp. 650--657.

[12] Victor Uc Cetina, "Multilayer Perceptrons with Radial Basis Functions as Value Functions in Reinforcement Learning," in *ESANN-European Symposium on Artificial Neural Networks-Advances in Computational Intelligence and Learning*, Burges, 2008.

[13] Damien Ernst, Bart De Schutter and Robert Babuška Lucian Buşoniu, "Continuous-State Reinforcement Learning with Fuzzy Approximation," vol. 4865, 2008.