

نقش یادگیری تفاوت زمانی تخلیص شده با تقریب زنده‌های تابع برای کاهش زمان و هزینه‌های محاسباتی یادگیری تقویتی

جواد عبدی^۱ اعظم فامیل خلیلی^۲ کارولوکس^۳ علی خاکی صدیق^۴

^۱ گروه مهندسی برق و کامپیوتر دانشکده فنی دانشگاه تهران - تهران - ایران
^۲ گروه مهندسی کامپیوتر دانشکده فنی دانشگاه آزاد اسلامی واحد کرج - کرج - ایران
^۳ مرکز تحقیقات فیزیک نظری ایران - تهران - ایران
^۴ دانشکده مهندسی برق دانشگاه خواجه نصیرالدین طوسی - تهران - ایران

کاهش هزینه‌های محاسباتی، با ترکیب $CMAC$ و TTD یادگیری سریع با کارایی محاسباتی و توانمندی‌های تعمیمی را بیان کند. نتایج تجربی ارائه شده، عملکرد موفقیت‌آمیز الگوریتم یادگیری Q را که با استفاده از دستورالعمل TTD و $CMAC$ در دو کار با فضاهای حالت پیوسته پیاده‌سازی شده‌اند، را نشان می‌دهد.

کلید واژه: یادگیری تقویتی، یادگیری تفاوت زمانی^۳، تفاوت زمانی تخلیص شده^۴، پاداش، تقریب، عامل‌های یادگیر، بروزآوری

۱- مقدمه

الگوریتم‌های یادگیری تقویتی [7][13][8] بر مبنای روش‌های تفاوت زمانی [14] در مورد تعامل اصلی این موضوع بین عامل یادگیرنده و محیط پیرامونی او بکار

چکیده - ترکیب الگوریتم یادگیری تقویتی^۱ با تقریب زنده‌های تابعی^۲ برای تعمیم فضای حالت اخیراً از توجه ویژه‌ای برخوردار شده و به صورت گسترده‌ای این اعتقاد وجود دارد که یکی از موارد تعیین کننده برای سنجش میزان یادگیری تقویتی به قلمروهای جالب عملی، است. این مقاله ترکیب دستورالعمل TTD ، یک پیاده‌سازی تقریبی مؤثر محاسباتی از روش‌های $TD(\lambda)$ با $CMAC$ ، یک تقریب‌گر تابع به ویژه مناسب برای یادگیری تقویتی درخور کارایی محاسباتی آن و توانایی یادگیری پیوسته را مورد بررسی قرار می‌دهد. اکثر مطالعات قبلی ترکیب $CMAC$ با الگوریتم‌های براساس $TD(0)$ که معمولاً برای $\lambda > 0$ یادگیری بسیار آهسته‌تر صورت می‌گیرد، یا با پیاده‌سازی مرسوم $TD(\lambda)$ که بر اساس آثار صلاحیت است به همراه هزینه‌های محاسباتی بالا مورد بررسی قرار داده‌اند. مطالعه فعلی، سعی دارد علاوه بر معرفی کامل روش TTD برای

³ Temporal Difference (TD)

⁴ Truncated Temporal Difference (TTD)

¹ Reinforcement Learning (RL)

² Function Approximators



مورد نظر می‌باشد.

$$z_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \quad (3)$$

که برگشت TD برای زمان t نامیده می‌شود. علائم روزرسانی که در معادله (۲) بکار رفته است، به این معنی است که مقدار تابع U برای بحث x_t می‌بایست با استفاده از مقدار خطا که در فرمول (۴) نشان داده شده است، تنظیم گردد. یعنی، به طرف $U_t(x_t) + \Delta$ ، به یک درجه‌ای که توسط یک نرخ یادگیری β کنترل می‌شود، کشیده شود.

$$\Delta = r_t + \gamma U_t(x_{t+1}) - U_t(x_t) \quad (4)$$

$$\text{update}^\beta(Q, x_t, a_t, r_t + \gamma \max_a Q_t(x_{t+1}, a) - Q_t(x_t, a)) \quad (5)$$

که در آن مقدار Q نسبت داده شده به هر زوج حالت-عمل (x, a) ، به منظور پیش‌بینی مقدار تنزیل داده شده تقویت دریافت شده بعد از اجرای عمل a در حالت x و سپس تعقیب یک سیاست حریمانه برای رسیدن به مقادیر Q فعلی. وقتی که مقادیر Q بهینه یاد گرفته شود، یک سیاست حریمانه در ارتباط با آنها، یک سیاست بهینه است.

در واقع، قانون عمومی ارائه شده در معادله (۲) با ساده‌ترین شکل روش‌های تفاوت زمانی $TD(0)$ مطابقت دارد. برای مقادیر کلی از ضریب λ جدید قانون به روز آوری $TD(\lambda)$ ، برای هر حالت x در گام زمانی t توسط فرمول زیر اعمال می‌شود:

$$\text{update}^\beta(U, x, (r_t + \gamma U_t(x_{t+1}) - U_t(x_t)) e_x(t)) \quad (6)$$

که در آن:

$$e_x(t) = \sum_{k=0}^t (\gamma \lambda)^{t-k} \chi_x(k) \quad (7)$$

مقدار اثر شایستگی^۲ [11][3] برای هر حالت x در زمان t و $\chi_x(t)$ مساوی ۱ است اگر $x = x_t$ و در غیر اینصورت صفر باشد. آثار شایستگی برای تمام حالات در هر

می‌رود. در هر گام زمانی t ، عامل حالت فعلی محیط، x_t ، را مشاهده می‌کند و عمل a_t را انجام می‌دهد. سپس یک ارزش یا پاداش تقویتی r_t دریافت می‌دارد، و حالت محیط به x_{t+1} تغییر می‌یابد. منظور از یادگیری شناخت بدون هیچ دانش قبلی از محیط است، یک سیاست تصمیم‌گیری (به عنوان مثال، یک ترسیم از حالات به عمل‌ها) که منجر به افزایش مقدار تنزیل داده شده تقویت، که عامل در طول دوره زندگی‌اش دریافت می‌نماید، می‌گردد.

$$E \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \quad (1)$$

که در آن $\gamma \in [0, 1]$ یک عامل تنزیل است که اهمیت نسبی پاداش‌های بلند مدت را نسبت به پاداش‌های کوتاه مدت تنظیم می‌کند.

۲- الگوریتم‌های بر مبنای الگوریتم یادگیری TD

گسترده‌ترین مطالعات الگوریتم‌های یادگیری تقویتی بر اساس روش تفاوت زمانی ساتن^۱ [14]، $TD(\lambda)$ است. این الگوریتم‌ها الزاماً به یادگیری توابع مقدار معین از حالت‌ها یا زوجی از حالت-عمل متکی هستند که کاربرد آنها را نسبت به پاداش‌های تنزیلی آینده برآورد می‌نماید. سیاست عامل یا مطلقاً از این ارزیابی‌ها استخراج می‌شود و یا قطعاً و صراحتاً از طریق یک تابع خط مشی ارائه می‌گردد. قاعده عمومی یادگیری تقویتی بر مبنای TD ممکن بصورت زیر نوشته شود:

$$\text{update}^\beta(U, x_t, r_t + \gamma U_t(x_{t+1}) - U_t(x_t)) \quad (2)$$

که U تابع کاربرد حالت است که به هر حالت، x یک برآورد از مقدار تنزیل داده شده تقویت دریافت شده که از حالت x شروع و سیاست جاری را دنبال می‌نماید. به ویژه آنکه، $U(x_t)$ برای پیش‌بینی مجموع تنزیلی پاداش‌های آتی

² Eligibility Traces

¹ Sutton

گام زمانی بر طبق قاعده به روزآوری زیر، بروز¹ می گردند:

$$e_x(t) = \gamma \lambda e_x(t-1) + \chi_x(t) \xrightarrow{e_x(0)=\chi_x(0)} \text{update}^\beta(U, x, (r_t + \gamma U_t(x_{t+1}) - U_t(x_t)) e_x(t)) \quad (8)$$

که در آن به صورت قراردادی $e_x(0) = \chi_x(t)$ می باشد.

توجه داشته باشید که برای $\lambda = 0$ فقط یک حالت x_t در گام زمانی t مقدار غیر صفر دارد و معادله شماره (6) به معادله (2) خلاصه می شود. برای λ مثبت، فرد می بایست در هر گام زمانی پیش بینی ها و آثار شایستگی را برای تمام حالات بروز در آورد. به همین دلیل است که این پیاده سازی با استفاده از $\lambda > 0$ از نظر محاسباتی بسیار گرانتر از وقتی که $\lambda = 0$ مورد استفاده قرار می گیرد می باشد، به ویژه برای کارهایی با فضای حالت بزرگتر. در هر صورت، استفاده از λ مثبت اغلب به یادگیری منجر می گردد که به صورت قابل ملاحظه ای سریعتر است. [8][14][15] تکنیک ساده تری بنام رویه TTD وجود دارد [6] که استفاده از λ عمومی با هزینه های محاسباتی پایین را میسر می نماید. این تکنیک به اختصار در بخش 3 توضیح داده خواهد شد.

3- استفاده از تقریب زنده های تابع

برای پیاده سازی هر الگوریتم یادگیری تقویتی بر مبنای تفاوت زمانی، فرد می بایست تعدادی روش ارائه تابع را انتخاب نموده و تابع بروزآوری را بطور مناسب تعریف نماید. ساده ترین انتخاب یک ارائه فهرست وار است که مقادیر تابع در یک جدول جستجو² ذخیره می شود و برای هر مقدار یک مدخل مجزا وجود دارد. به روز رسانی مقدار Q برای حالت x و عمل a با مقدار خطای Δ و میزان یادگیری β به سادگی به صورت زیر پیاده سازی می شود.

$$Q(x, a) := Q(x, a) + \beta \Delta \quad (9)$$

اکثر نتایج نظری مربوط به همگرایی الگوریتم های

یادگیری تقویتی یک ارائه فهرست وار را فرض می کنند، ولی این شیوه برخی وقفه های عملی مهم را نیز دارد. شیوه ارائه جدولی به سختی باری کارهایی با فضاهای حالت بزرگ کاربرد دارد، نه تنها به خاطر احتیاجات حافظه ای فزاینده بلکه اول از همه، برای زمان بسیار طولانی برای همگرایی. برای کارهایی با فضاهای حالت پیوسته، این ارائه به هیچ وجه به طور مستقیم کاربرد ندارد.

باور عمومی بر آنست که استفاده از روش های تقریب تابع به منظور نمایش تابع های مقدار در الگوریتم های RL نه تنها ممکن است حافظه را ذخیره کند بلکه تعمیم بر فضای حالت را نیز میسر می نماید و بنابراین کاربرد موفق این الگوریتم ها در کارهای بزرگ را ممکن می سازد. نتایج نظری اخیر وجود دارد که درک ما از مزایای بالقوه و محدودیت های استفاده از تقریب تابع برای یادگیری تقویتی را بهبود می بخشد. [2][7][12][16] همچنین تعدادی مطالعات تجربی با ترکیب های یادگیری تقویتی و چندین تقریب زنده تابع، شامل شبکه پرسپترون چند لایه [8] تقریب زنده های بر مبنای حافظه، [11] یا $CMAC$ [17][15]، که هم موفقیت آمیز و هم ناامید کننده بوده اند، وجود داشته است. از تعداد زیادی تقریب زنده² آزمایش شده با RL تا این زمان، $CMAC$ به علت سرعت یادگیری بالای آن، کارآئی محاسباتی و توانائی یادگیری پیوسته، بطور ویژه ای مورد توجه قرار گرفته است. این مسئله در بخش 4 به اختصار توضیح داده می شود و در ترکیب با دستورالعمل TTD در آزمایش های ارائه شده در بخش 5 مورد استفاده قرار می گیرد.

4- تفاوت های زمانی تخلیص شده

این قسمت به تفصیل توضیح می دهد که چگونه رویه TTD می تواند از یک فرمولاسیون متفاوت $TD(\lambda)$ مشتق شود و چگونه می تواند برای پیاده سازی الگوریتم های یادگیری

¹ Update

² Lookup Table

$$\Delta_t^\lambda = \sum_{k=0}^{\infty} (\gamma\lambda)^k [r_{t+k} + \gamma U_{t+k}(x_{t+k+1}) - U_{t+k}(x_{t+k})] \quad (13)$$

$$= \sum_{k=0}^{\infty} (\gamma\lambda)^k \Delta_{t+k}^0$$

همان طوری که در [5][18] نشان داده شده، مجموع خطای $TD(\lambda)$ برای حالت x بر حسب خطاهای Δ_t^λ به صورت زیر بیان می‌شود:

$$\sum_{t=0}^{\infty} \Delta_x(t) = \sum_{t=0}^{\infty} \Delta_t^\lambda \chi_x(t) \quad (14)$$

بنابراین $\Delta_t(t)$ در هر گام زمانی t برای هر x محاسبه، و برای بروزرسانی $U(x)$ به ازای هر x بکار برده می‌شود، و Δ_t^λ در هر گام زمانی t فقط برای x_t محاسبه می‌شود و فقط برای تعیین $U(x_t)$ بکار برده می‌شود، تاثیرات کلی استفاده از این دو فرم از خطاهای TD یکسان هستند. بازگشت $TD(\lambda)$ برای زمان t به صورت زیر تعریف می‌شود: [17]

$$z_t^\lambda = (1-\lambda) \sum_{k=0}^{\infty} \lambda^k z_t^{(k+1)} \quad (15)$$

$$= \sum_{k=0}^{\infty} (\gamma\lambda)^k [r_{t+k} + \gamma(1-\lambda)U_{t+k}(x_{t+k+1})]$$

مجموع پاداش فوری¹ و فایده پیش‌بینی شده² تنزیل حالت جانشین برای $\lambda=0$ ساده است. برای $\lambda=1$ بازگشت TD معمول، مجموع تنزیل کلی تقویت، برقرار است. استفاده از λ در بازه $0 < \lambda < 1$ اجازه می‌دهد که به آسانی بین این دو حد تعریف شود. باید به خاطر داشت که تعریف بالا ممکن است به طور بازگشتی به صورت زیر نیز بیان شود:

$$z_t^\lambda = r_t + \gamma(\lambda z_{t+1}^\lambda + (1-\lambda)U_t(x_{t+1})) \quad (16)$$

واضح است که اگر اندیس‌های زمان حذف شود، خواهیم داشت:

$$\Delta_t^\lambda = z_t^\lambda - U(x_t) \quad (17)$$

تقویت شده بر مبنای TD بکار برده شود. همچنین تحلیل ساده‌ای از مزایای کارآیی محاسباتی آن در مقایسه با اثرهای شایستگی مطرح می‌شود. [6][18]

۴-۱- خطاهای TD و بازگشت‌های TD

اگر یک رشته از حالت‌ها را به صورت $x_0, x_1, \dots, x_k, \dots$ و همچنین مقادیر سیگنال تقویت نسبت داده شده به آنها را به ترتیب به صورت $r_0, r_1, \dots, r_k, \dots$ در نظر بگیریم، و با توجه به فرمول تعمیم یافته (۴)، برای حالت x_t می‌توان نوشت:

$$\Delta_x = \sum_{t=0}^{\infty} \Delta_x(t) = \sum_{t=0}^{\infty} \left\{ (r_t + \gamma U_t(x_{t+1}) - U_t(x_t)) \sum_{k=0}^{\infty} (\gamma\lambda)^{t-k} \chi_x(k) \right\} \quad (10)$$

و

$$\begin{aligned} \Delta_{xt} &= r_t + \gamma U_t(x_{t+1}) - U_t(x_t) + \\ &= \gamma\lambda [r_{t+1} + \gamma U_{t+1}(x_{t+2}) - U_{t+1}(x_{t+1})] \\ &= (\gamma\lambda)^2 [r_{t+2} + \gamma U_{t+2}(x_{t+3}) - U_{t+2}(x_{t+2})] + \dots \\ &= \sum_{k=0}^{\infty} (\gamma\lambda)^{t-k} (r_{t+k} + \gamma U_{t+k}(x_{t+k+1}) - U_{t+k}(x_{t+k})) \end{aligned} \quad (11)$$

اگر یک حالت چندین مرتبه در دنباله اتفاق بیفتد، هر حالت تکراری به یک صورت مشابه بروز می‌شود. این مشاهده ساده راهی به سوی فرمولاسیون تناوبی $TD(\lambda)$ می‌گشاید که امکانات پیاده‌سازی جدیدی را میسر می‌سازد. با فرض:

$$\Delta_t^0 = r_t + \gamma U_t(x_{t+1}) - U_t(x_t) \quad (12)$$

در فرمول (۱۲) خطای $TD(0)$ در گام زمانی t نشان داده شده است.

خطای $TD(\lambda)$ در زمان t با استفاده از خطاهای $TD(0)$ به صورت زیر تعریف می‌شود:

¹ Immediate Reward

تصحیح برای $\gamma(1-\lambda)$ های بزرگ، این اطلاعات تقریباً دست نیافتنی بودند.

۴-۳- رویه TTD

همانطور که توضیح داده شد، خطاهای $TD(\lambda)$ تلخیص شده m مرحله‌ای، بوسیله پیگیری آخرین m حالت ملاقات شده و بروزرسانی فایده پیش‌بینی شده آخرین حالت یک بافر تجربه m -عنصری حاصل شده با رکوردهای $\langle x_{t-k}, a_{t-k}, r_{t-k}, U_{t-k}(x_t - k + 1) \rangle$ است که برای همگی آنها $k = 0, 1, \dots, m-1$ و t گام زمانی جاری است. در هر گام t با نوشتن $r_{[k]}, a_{[k]}, x_{[k]}$ به عناصر مورد نظر بافری که $x_{t-k}, a_{t-k}, r_{t-k}$ و $U_{t-k}(x_t - k + 1)$ را ذخیره کرده، رجوع می‌شود. ارجاع‌های U با گام‌های زمانی اندیس گذاری نشده‌اند، چون همگی آنها مقادیر موجود در هر گام زمانی جاری را نشان می‌دهند. با این توضیحات، عملکرد رویه $TTD(\lambda, m)$ در نمودار (۱) نشان داده شده است.

نمودار ۱: رویه دستورالعمل $TTD(\lambda, m)$

در هر گام زمانی t	
(۱)	حالت فعلی x_t مشاهده می‌شود؛ $x_{[0]} := x_t$
(۲)	یک عمل a_t برای حالت x_t انتخاب می‌شود. $a_{[0]} := a_t$
(۳)	عمل a_t انجام می‌شود، حالت جدید x_{t+1} پاداش فوری r_t را حاصل می‌کند.
(۴)	$r_{[0]} := r_t; u_{[0]} := U(x_{t+1})$
(۵)	$z := ttd_return(0)$
(۶)	$update^\beta(U, x_{[m-1]}, a_{[m-1]}, z - U(x_{[m-1]}))$
(۷)	بروزآوری. اندیس‌های بافر تجربه جابجا می‌شود.

این معادله نشان می‌دهد که چگونه در یادگیری بازگشت‌های $TD(\lambda)$ می‌توانند به جای خطاهای $TD(\lambda)$ به کار روند. این تنها برای حالت یادگیری دسته^۱ درست می‌شود، اما در واقع هدف اصلی یادگیری تقویتی مبتنی بر TD ، انعکاس پیش‌بینی درست بازگشت‌های TD می‌باشد.

۴-۲- بازگشت‌های $TD(\lambda)$ تلخیص شده

متأسفانه در عمل تا زمانی که بازگشت‌های $TD(\lambda)$ موجود نیستند، مشاهده شرح داده شده در فرمول (۱۷) نمی‌تواند مستقیماً بکار برده شود. در [5] پیشنهاد شده است که آنها را با بازگشت‌های $TD(\lambda)$ تلخیص شده به صورت زیر تقریب بزنند:

$$\begin{aligned} z_t^{\lambda, m} &= \sum_{k=0}^{m-2} (\gamma\lambda)^k [r_{t+k} + \gamma(1-\lambda)U_{t+k}(x_{t+k+1})] \\ &\quad + (\gamma\lambda)^{m-1} [r_{t+m-1} + \gamma U_{t+m-1}(x_{t+m})] \quad (18) \\ &= \sum_{k=0}^{m-1} (\gamma\lambda)^k [r_{t+k} + \gamma(1-\lambda)U_{t+k}(x_{t+k+1})] \\ &\quad + (\gamma\lambda)^m U_{t+m-1}(x_{t+m}) \end{aligned}$$

که، $z_t^{\lambda, m}$ بازگشت $TD(\lambda)$ مختصر شده m مرحله‌ای یا بازگشت $TTD(\lambda, m)$ در زمان t نامیده می‌شود. باید به خاطر داشت که $z_t^{\lambda, m}$ تعریف شده توسط فرمول (۱۸) تصحیح شده است، بعنوان مثال $z_t^{\lambda, m}$ را نمی‌توان به آسانی از فرمول مختصر شده (۱۵) بدست آورد. اصطلاح تصحیح شده فرمول مختصر شده $(\gamma\lambda)^m U_{t+m-1}(x_{t+m})$ منجر به ضرب شدن آخرین پیش‌بینی $U_{t+m-1}(x_{t+m})$ در γ ، به جای ضرب شدن در $\gamma(1-\lambda)$ می‌شود که بطور مجازی معادل استفاده از $\lambda = 0$ برای آن مرحله است، و به همین ترتیب برای $U_{t+m-1}(x_{t+m})$ برای گام‌های زمانی $(t+m, t+m+1, \dots)$ محاسبه می‌شود. بدون این

^۱ Batch Learning Mode

$$z_t^{\lambda,m} = S_t^{\lambda,m} + T_t^{\lambda,m} + W_t^{\lambda,m} \quad (19)$$

که

$$S_t^{\lambda,m} = \sum_{k=0}^{m-1} (\gamma\lambda)^k r_{t+k}$$

$$T_t^{\lambda,m} = \sum_{k=0}^{m-2} (\gamma\lambda)^k \gamma(1-\lambda)U_{t+k}(x_{t+k+1}) \quad (20)$$

$$W_t^{\lambda,m} = (\gamma\lambda)^{m-1} \mathcal{W}_{t+m-1}(x_{t+m})$$

که $W_t^{\lambda,m}$ می‌تواند مستقیماً برای هر m ای در زمان ثابت

محاسبه شود. لذا کاملاً مفروض است که:

$$\begin{cases} S_{t+1}^{\lambda,m} = \frac{1}{\gamma\lambda} [S_t^{\lambda,m} - r_t + (\gamma\lambda)^m r_{t+m}] \\ T_{t+1}^{\lambda,m} = \frac{1}{\gamma\lambda} [T_t^{\lambda,m} - \gamma(1-\lambda)(U_t(x_{t+1}) - W_{t+1}^{\lambda,m})] \end{cases} \quad (21)$$

فرمول فوق الگوریتم محاسبه فزاینده $S_t^{\lambda,m}$ و در نتیجه محاسبه $z_t^{\lambda,m}$ در زمان ثابت (کوچک) برای هر m دلخواه را توضیح می‌دهد. این الگوریتم از نظر ریاضی کاملاً معادل روش تناوبی نشان داده شده در بالا، است.¹

TTD فزاینده امکان استفاده $TD(\lambda > 0)$ را با هزینه‌ای کاملاً برابر هزینه محاسباتی $TD(0)$ ، ممکن می‌سازد.

۴-۳-۲- انتخاب m

انتخاب منطقی m ، به وضوح به λ بستگی دارد. برای $\lambda = 0$ بهترین امکان $m = 1$ است و برای $\lambda = 1$ و $\gamma = 1$ هیچ مقدار محدودی از m وجود ندارد که برای تقریب زدن دقیق $TD(\lambda)$ به اندازه کافی بزرگ باشد. خوشبختانه، نتایج عملی موجود گذشته در مورد $TD(\lambda)$ نشان می‌دهد که معمولاً $\lambda = 1$ مقدار بهینه‌ای برای استفاده نیست. [8][14][17][19]

¹ ضرورتاً از حیث عددی معادل نیستند، زیرا ممکن است در بعضی پیاده سازی‌های عملی ایجاد مشکل کند.

در ابتدای یادگیری، قبل از اینکه اولین m گام‌ها بوجود آید، هیچ یادگیری نمی‌تواند اتفاق بیفتد. در جریان گام‌های اولیه عملکرد رویه TTD برای بروز کردن محتویات بافر تجربه به نحوی شایسته، کاهش می‌یابد. در نمودار (۱) برای سادگی کار از این جزئیات تکنیکی چشم‌پوشی شده است و در ادامه نیز از آنها صرف نظر شده است.

۴-۳-۱- محاسبات بازگشت TTD

عملکرد مرحله ۵ الگوریتم، یعنی $ttd_return(k_0)$ بازگشت TTD را در زمان $t - m + 1$ ، بر مبنای محتویات ذخیره شده در بافر تجربه با اندیس‌های از k_0 تا $m - 1$ محاسبه می‌کند. نمودار (۱)، مقدار $k_0 = 0$ را بکار می‌برد. اما همانطور که بعداً دیده می‌شود در بعضی موارد $k_0 > 0$ نیز نیاز است.

در [5] دو روش برای اجرای این محاسبه پیشنهاد شده است، یکی روش تناوبی ساده و دیگری روش فزاینده‌ای، که از نظر محاسباتی کارآمدتر است. روش تناوبی بر مبنای کاربردهای تکراری فرمول (۱۶) است:

For $k = k_0, k_0 + 1, \dots, m - 1$ do

if $k = k_0$ then $z := r_{[k]} + \gamma u_{[k]}$

else $z := r_{[k]} + \gamma(\lambda z + (1 - \lambda)u_{[k]})$

هزینه محاسباتی چنین تکنیری، در صورت بازگشت بموقع، برای مقادیر منطقی m قابل قبول است. برای بعضی از روش‌های ارائه تابع، مثل شبکه‌های عصبی، رویهمرفته پیچیدگی زمانی در مقابل هزینه‌های بازیابی و بروزرسانی مقدار یک تابع که در مرحله ۴ و ۶ انجام می‌شود، و همچنین هزینه محاسباتی z قابل صرف نظر کردن است.

برای نشان دادن چگونگی انجام این محاسبه به طور فزاینده، تعریف بازگشت TTD (فرمول (۱۸)) به دو قسمت تقسیم می‌شود:

طرفی دیگر، m های خیلی بزرگ ممکن است برای تعریف وقفه‌های طولانی بین تجربه و یادگیری بکار برده شوند که اثرات زیان باری در پی خواهد داشت اگر چه این پدیده خارق العاده تا حالا مشاهده نشده است. در قسمت مطالعات تجربی مقادیر مختلف m برای λ در بازه صفر تا یک مورد ارزیابی قرار گرفته است.

جدول ۱: انتخاب m : یک ارائه

$\gamma\lambda$	0.99	0.975	0.95	0.9	0.8	0.6
$\min\left\{m \mid (\gamma\lambda)^m < \frac{1}{10} \gamma\lambda\right\}$	231	92	46	23	12	6

الگوریتم مورد نظر، که در نمودار (۲) نشان داده شده است به عنوان جانشینی برای الگوریتم اصلی نمودار (۱) برای گام زمانی پایانی، فرموله شده است.

نمودار ۲: راه‌اندازی مجدد برای رویه دستورالعمل

$$TTD(\lambda, m)$$

در گام زمانی پایانی t :

(۱) حالت فعلی x_t مشاهده می‌شود؛ $x_{t[0]} := x_t$

(۲) یک عمل a_t برای حالت x_t انتخاب می‌شود.

$$a_{t[0]} := a_t$$

(۳) عمل a_t انجام می‌شود، حالت جدید x_{t+1} پاداش فوری r_t را حاصل می‌کند.

$$r_{t[0]} := r_t; u_{t[0]} := U(x_{t+1}) \quad (۴)$$

(۵) برای $k = 0, 1, \dots, m-1$ انجام می‌دهد:

(الف) برای $k = k_0, k_0 + 1, \dots, m-1$ انجام می‌دهد:

$$z := r_{[k]} + \gamma u_{[k]} \quad \text{اگر } k = k_0$$

$$z := r_{[k]} + \gamma(\lambda z + (1-\lambda)u_{[k]}) \quad \text{دیگر}$$

(ب) $update^\beta(U, x_{[m-1]}, a_{[m-1]}, z, U(x_{[m-1]}))$

(ج) اندیس‌های بافر تجربه جابجا می‌شود.

برای $\lambda < 1$ یا $\gamma < 1$ ترجیحاً مقداری از m مورد نظر است که تنزیل $(\gamma\lambda)^m$ ، عدد کوچکی شود. یک تعریف ممکن از کوچک این است که مثلاً کمتر از $\gamma\lambda$ باشد. جدول (۱) تاثیرات تجربی این ابتکار را نشان می‌دهد. این معیار غیر رسمی کاملی است اما مطالعات عملی [20][18][5] نشان داد که حتی m های نسبتاً کوچک نیز خوب عمل می‌کنند. از

۴-۳-۳- عملیات راه‌اندازی مجدد

در فعالیت‌های پراکنده^۱ [13] و بسیاری فعالیت‌های دنیای واقعی، بعضی زمان‌های یادگیری باید متوقف شود و این امر ضرورت طراحی یک مکانیزم ویژه برای رویه TTD را اجباری می‌کند، که عملیات راه‌اندازی مجدد نامیده می‌شود. عملیات راه‌اندازی مجدد در پایان هر رویداد از رویدادهای نامنظم، یا بعد از اتمام یادگیری فراخوانده می‌شود. تنها مسأله‌ای که باید مورد بحث قرار گیرد این است که بافر تجربه شامل ذخیره‌ای از m مرحله آخر است که هنوز یادگیری برای آنها انجام نشده و نیز هیچ مرحله دیگری هم وجود نخواهد داشت که یادگیری را برای مراحل باقیمانده ممکن سازد. راه حل واضح این است که m مرحله خیالی اضافی را به طریقی که یادگیری را برای تمام مراحل واقعی باقی مانده در بافر و بازگشت‌های TTD ای که تحت تاثیر مراحل خیالی شبیه‌سازی شده قرار نگرفته‌اند، ممکن است، شبیه‌سازی نمود.

^۱episodic

پیاده‌سازی اثرات شایستگی مستلزم این است که عملیات‌های زیر در هر مرحله انجام شوند:

(۱) بروزرسانی اثرات شایستگی در هر حالت،
(۲) ذخیره‌سازی بازایی فواید پیش‌بینی شده حالت جاری و جانشین‌های آن،

(۳) محاسبه مقدار خطای $TD(0)$ جاری،

(۴) بروزرسانی فواید پیش‌بینی شده تمام حالت‌های با اثرات شایستگی غیر صفر، یعنی در بدترین حالت، تمام n حالت را بروز نماید^۱. هزینه‌های این عملیات‌ها به ترتیب عبارتند از: $2c_+ + c_*$ ، $2c_r$ ، $(N-1)c_* + c_* + c_+$ و $N(c_+ + c_u)$ ، که روی هم رفته هزینه بدبینی پیاده‌سازی اثرات شایستگی در هر مرحله برابر خواهد شد با:

$$C(ET) = [(N-1)c_* + c_* + c_+] + 2c_r + [2c_+ + c_*] + N(c_* + c_u) \quad (22)$$

$$= 3c_+ + (2N+1)c_* + 2c_r + Nc_u$$

رویه TTD در هر مرحله فایده پیش‌بینی شده حالت فعلی را ذخیره‌سازی می‌کند، بازگشت‌های $TD(\lambda)$ مختصر شده را محاسبه می‌کند. فایده پیش‌بینی شده آخرین حالت در بافر تجربه را ذخیره‌سازی می‌کند. خطای TTD آن حالت را بصورت اختلاف بازگشت TTD و فایده آن حالت محاسبه می‌کند، و در نهایت فایده پیش‌بینی شده آن حالت را بروز می‌کند.

طبق الگوریتم تناوبی ساده، هزینه محاسبه بازگشت TTD برابر است با $c_+ + c_* + (m-1)(2c_+ + 2c_*)$ و هزینه‌های باقی عملیات‌ها به ترتیب عبارتند از: c_r ، c_+ و c_u . بنابراین هزینه کلی در رویه تناوبی TTD در هر گام زمانی برابر است با:

$$C(TTD_{iter}) = [c_+ + c_* + (m-1)(2c_+ + 2c_*)] + c_r + c_r + c_+ + c_u \quad (23)$$

$$= 2mc_+ + (2m-1)c_* + 2c_r + c_u$$

این بخش تحلیل ساده‌ای از پس اندازه‌های محاسباتی با استفاده از رویه TTD ، در مقایسه با اثر شایستگی ارائه می‌کند. هزینه عملیات یک مرحله‌ای پیاده‌سازی اثرات شایستگی $TD(\lambda)$ که در فرمول (۸) تعریف شد و همچنین الگوریتم TTD پایه‌ای در نمودار (۱) نشان داده شده است. تحلیل ارائه شده به طور مناسب می‌تواند برای هر الگوریتم یادگیری تقویتی بر مبنای TD پیاده‌سازی شود. معمولاً فرض می‌شود که از یک ارائه تابعی جدولی^۱ استفاده می‌شود، هر چند که ممکن است بعضی نتایج برای دیگر ارائه‌ها نیز بخوبی معتبر باشند.

نمادهای زیرین برای تعیین هزینه‌های عملیات‌های اصلی بکار برده می‌شوند.

C_+	هزینه عملیات‌های افزودنی
C_*	هزینه عملیات‌های صفر
C_r	هزینه ذخیره‌سازی مقدار یک تابع
C_u	هزینه بروزرسانی مقدار یک تابع

و نماد N نشان دهنده سائز مؤثر فضای حالت است، مثلاً تعداد کل حالت‌های متمایز که برای فواید پیش‌بینی شده یاد گرفته شده‌اند^۲.

فرض، که از یک λ با مقدار ثابت استفاده می‌شود، این امر باعث ایجاد زیر عبارت‌هایی مثل λ یا $\gamma(1-\lambda)$ می‌شود که می‌توانند از قبل محاسبه شوند و سپس به عنوان مقادیر ثابت بکار برده شوند. برای سادگی کار، از هزینه‌های عملیات‌های اختصاص دادن‌ها و مقایسه‌ها صرف نظر می‌شود.

^۱ Tabular Function

^۲ در واقع سائز فضای حالت است اما برای فضاهای حالت پیوسته، N ممکن است نشان دهنده تمامی نواحی باشد که فضای حالت از طریق کوانتیزاسیون به آنها تقسیم شده است. در صورتیکه پیش‌بینی‌های در یک شبکه عصبی ذخیره شده باشند، N ، تعداد وزنه‌های شبکه است.

و هزینه‌های TTD کمتر از هزینه‌های TTD کمتر از هزینه‌های اثرات شایستگی است اگر و تنها اگر، در روش تناوبی،

$$12m + 3 < 16N + 10 \equiv m < \frac{4}{3}N + \frac{7}{12} \quad (۳۳)$$

و در روش افزایشی،

$$34 < 16N + 10 \equiv N > \frac{3}{2} \quad (۳۴)$$

در فعالیت‌های واقعی‌تر می‌توان انتظار داشت که $m \ll N$. یعنی شرایط داده شده در فرمول (۲۹) یا (۳۱) باید همراه برقرار باشند، یعنی حتی ساده‌ترین رویه TTD تناوبی باید کارآمدتر از اثرات شایستگی باشد. هر چند در حالت‌های قطعی، این شرایط ممکن است معتبر نباشند. آنها از این فرض ساده‌سازی مشتق شده‌اند که تمام N حالت در هر مرحله مقادیر اثرات شایستگی غیر صفر دارند و این بطور آشکارا خیالی است، البته مواقعی وجود دارد که تنها n حالت با اثرات شایستگی غیر صفر وجود دارد که $n \ll N$. اگر $n \approx m$ باشد، مزیت کارآیی TTD تناوبی بدیهی نخواهد بود.

باید بخاطر داشت که هزینه تابعی که عملیات را بروز می‌کند، c_{ii} ، در عمل ممکن است خیلی بزرگتر از هزینه‌های عملیات‌های اضافی و ضربی، c_+ و c_* ، باشد و بنابراین انتظار می‌رود که TTD سودمند و با صرفه است. حتی برای روش‌های نمایش تابع تخمین پارامتری، مثل شبکه‌های عصبی، که پیاده‌سازی اثرات شایستگی آنها بطور خاصی مناسب هستند، نیز ضروری است که تمام وزن‌ها و اثرات شایستگی‌شان را در هر مرحله بروز نماید و این در حالی است که TTD ای که از نمایش تابع یکسانی استفاده می‌کند در هر مرحله بازگشت TTD را محاسبه می‌کند و اگر تعداد وزن‌ها در مقایسه با m بیشتر باشد تنها وزن‌ها را بروز می‌کند که ممکن است به طور منطقی و از نظر محاسباتی ارزاتر باشند.

استفاده از روش افزایشی برای محاسبه بازگشت TTD می‌تواند هزینه محاسباتی رویه TTD را کاهش دهد. در این روش هزینه محاسبه بازگشت TTD در هر مرحله برابر است با مجموع $3c_* + 4c_+$ (که عبارت $S_i^{\lambda,m}$ را بروز می‌کند)، c_* (که عبارت $T_i^{\lambda,m}$ را بروز می‌کند)، و c_+ (جمع دو عبارت). پس هزینه کلی TTD به صورت زیر خواهد شد:

$$C(TTD_{iter}) = c_r + [3c_* + 4c_+ + c_* + c_+] \quad (۲۴)$$

$$+ c_r + c_+ + c_{ii}$$

$$= 6c_+ + 4c_* + 2c_r + c_{ii}$$

که دیگر به m بستگی ندارد.

با کلی ساده سازی بیشتر و جایگزینی تمام هزینه‌های مقدماتی با مقدار ثابت c_* ، خواهیم داشت:

$$C(ET) = (3N + 6)c \quad (۲۵)$$

$$C(TTD_{iter}) = (4m + 2)c \quad (۲۶)$$

$$C(TTD_{iner}) = 13c \quad (۲۷)$$

پس، روش تناوبی رویه TTD از نظر محاسباتی ارزاتر از پیاده‌سازی اثرات شایستگی است، اگر و تنها اگر:

$$4m + 2 < 3N + 6 \equiv m < \frac{3}{4}N + 1 \quad (۲۸)$$

در TTD افزایشی شرط متناظر به صورت زیر خواهد بود:

$$13 < 3N + 6 \equiv N > \frac{7}{3} \quad (۲۹)$$

برای رسیدن به تخمین واقع بینانه‌تر، فرض می‌شود که:

$$c_r = c_+ = c, \quad c_* = c, \quad c_{ii} = c_* + c_+ = 6c$$

$$c_{ii} = c_* + c_+ = 6c$$

$$C(ET) = (16N + 10)c \quad (۳۰)$$

$$C(TTD_{iter}) = (12m + 3)c \quad (۳۱)$$

$$C(TTD_{iner}) = 34c \quad (۳۲)$$

^۱ اگر تعداد زیادی حالت با اثرات شایستگی برابر صفر وجود داشته باشد، هزینه این پیاده سازی ممکن است به اندازه قابل توجهی کاهش یابد.

۵-CMAC: یک جدول جستجوی بد کد شده

CMAC (کنترل کننده بیان مدل مغزی) توسط آلباس در سال ۱۹۸۱ که قسمتی به عنوان مدلی از عملکرد مغز و قسمتی به عنوان روش مفید نگهداری تابع و تقریب، پیشنهاد شد. مزایای آن به طور وسیع تا پایان دهه هشتاد شناخته شده نبود، از آن زمان این روش بیشتر و بیشتر با موفقیت در کنترل خودکار بکار رفت. [9][1]

پیاده سازی ویژه CMAC بکار رفته در این مقاله کاملاً از توصیف اصلی آلباس پیروی می نماید. برای به دست آوردن ایده ای کلی از چگونگی کار آن، ممکن است به آن به عنوان یک جدول جستجوی بد کد شده پراکنده بنگریم که از پوشش های چندگانه روی هم برای پوشاندن فضای ورودی استفاده می کند، به نحوی که هر نقطه ورودی در شماری از پوشش های روی هم قرار می گیرد. پوشش ها نسبت به هم در فواصل زمانی ثابت جابجا می شوند به طوریکه فضای حالت به شکلی کارآمد به مناطق کوچک تقسیم می گردد، اندازه این تقسیمات قطعیت نمودار را تعیین می کند. هر پوشش، متناظر با یک عنصر از بردارهای وزن های CMAC می باشد.

فرض کنید تعداد k پوشش وجود داشته باشد که روی هم قرار گرفته اند و شامل بردار ورودی مفروض x و $t_1(x), t_2(x), \dots, t_K(x)$ باشند. هر یک از آنها می تواند به یک عنصر از بردار وزن های $k_1(x), k_2(x), \dots, k_K(x)$ متناظر شود. سپس تابع برای x می تواند به سادگی به عنوان مجموع این وزن ها محاسبه شود $w[k_1(x)] + w[k_2(x)] + \dots + w[k_K(x)]$. جهت بروزآوری مقدار تابع برای x با استفاده از مقدار خطای Δ و نرخ یادگیری β ، هر یک از وزن های متناظر k باید طبق قاعده زیر تنظیم گردند:

$$w[k] := w[k] + \frac{\beta}{K} \Delta \quad (37)$$

که در آن: $K = k_1(x), k_2(x), \dots, k_K(x)$

پس بنابراین واضح است که نسخه تناوبی رویه TTD کارآمدتر از اثرات شایستگی است. البته، با توجه به فرمول های (۲۹) و (۳۴)، اگر بازگشت TTD به طور فزاینده محاسبه شود، هزینه TTD برای N یا n های واقعی دلخواه همواره کمتر خواهد بود.

۴-۴- یادگیری Q مبتنی بر TTD

برای پیاده سازی الگوریتم های یادگیری تقویت شده مبتنی بر TD خاص که بر اساس رویه TTD باشند، فقط باید برای U مقادیر تابع مقتضی را جایگزین کرد، و عملیات روزرسانی مرحله ۶ در نمودار ۱ و مرحله ۵-ب در نمودار ۲ را تعریف نمود. بخصوص برای الگوریتم یادگیری Q باید:

(۱) در مرحله ۴ نمودار ۱ $U(x_{t+1})$ را با $\max_a Q(x_{t+1}, a)$ جایگزین کرد.

(۲) مرحله ۶ نمودار ۱ و مرحله ۵-ب نمودار ۲ را به صورت

زیر پیاده سازی نمود:

$$\text{update}^\beta(Q, x_{[m-1]}, a_{[m-1]}, z - Q(x_{[m-1]}, a_{[m-1]})) \quad (35)$$

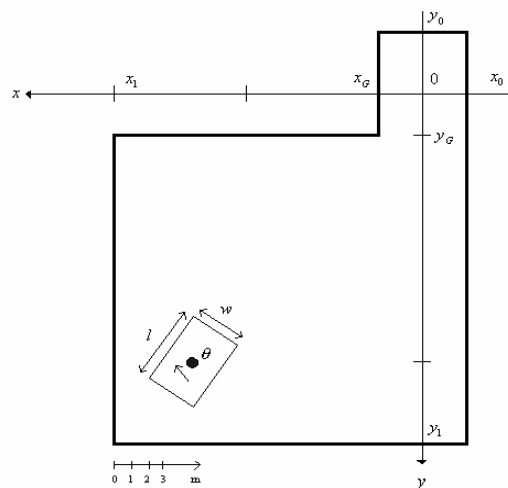
برای به دست آوردن فوریت کاری از الگوریتم یادگیری Q بر اساس TTD عمومی، در کنار تجزیه موارد ارائه تابع، فرد می بایست برخی مکانیزم های انتخاب تابع عمل را پیاده سازی نماید، تعیین کند که در هر گام زمانی کدام عمل بر اساس مقادیر فعلی Q اجرا شود. در این کار مکانیزم انتخاب احتمالی بر مبنای توزیع استاندارد بولتزمن مورد استفاده قرار گرفته است، که بر طبق آن احتمال انتخاب عمل a^* در حالت x مساوی است با:

$$\text{prob}(x, a^*) = \frac{\exp\left(\frac{Q(x, a^*)}{T}\right)}{\sum_a \exp\left(\frac{Q(x, a^*)}{T}\right)} \quad (36)$$

که در آن درجه حرارت $T > 0$ مقدار اتفاقی را تنظیم

می کند.

دارد. عامل ملزم به پارک کردن خودرو در گاراژ می‌باشد به نحویکه خودرو کاملاً در داخل آن قرار گیرد. مراحل این کار به صورت مرحله‌ای است و شامل شماری از آزمون‌های مستقل می‌شود. وقتی که خودرو وارد گاراژ می‌شود و یا وقتی که به دیوار برخورد می‌کند آزمون به انتها می‌رسد. بعد از هر امتحان خودرو دوباره به محل اولیه خود برمی‌گردد.



شکل ۲: مسئله پارک اتومبیل اندازه ابعاد شکل عبارتند از:
 $w = 2m, l = 4m, x_0 = -1.5m, x_G = -1.5m, x_G = 1.5m$
 $x_1 = 8.5m, y_0 = -3m, y_G = 3m, y_1 = 13m$

نمودار حالت شامل سه متغیر است: مختصات مستطیلی مرکز خودرو، x, y و زاویه θ بین محور خودرو و محور x از سیستم مختصات. محل استقرار اولیه خودرو ثابت بوده و بدین شکل توصیف می‌گردد:

$$x = 6.15m, y = 10.47m, \theta = 3.7rad$$

عمل‌های قابل پذیرش عبارتند از: "مستقیم برآیند" به چپ پیچید" به راست پیچید". اثرهای این اعمال، معادلات حرکت دقیق، و دیگر جزئیات در ضمیمه الف ارائه شده است. عامل هرگاه خودرو را با موفقیت در گاراژ پارک می‌کند یک تقویت با مقدار ۱ (یک پاداش) دریافت می‌کند و هرگاه به دیوار برخورد کند یک تقویت با مقدار ۱- (یک جریمه). در دیگر گام‌های زمانی تقویت صفر است.

تعداد وزن‌ها برای نگهداری تابع جمع شده‌اند و برای عملیات بروزآوری تابع تنظیم گردیده‌اند که ثابت و مساوی k می‌باشد، و معمولاً کسر بسیار کوچکی از کل تعداد وزن‌ها است. این امر، $CMAC$ را از نظر محاسباتی در مقایسه با اکثر شبکه‌های عصبی مصنوعی بسیار کارآمد می‌سازد. برای استفاده از روش ارائه تابع فوق با الگوریتم یادگیری Q ، طبیعی است که یک جدول مجزای $CMAC$ را به هریک از عمل‌های ممکن تخصیص می‌دهیم. هریک از این جدول‌های $CMAC$ حالاتی را به عنوان ورودی دریافت می‌نماید و به صورت فزاینده مقادیر Q را برای عمل متناظر و تمام حالات یاد می‌گیرد.

۶- مطالعات تجربی

مطالعات تجربی با ترکیب الگوریتم یادگیری Q بر مبنای TTD و $CMAC$ انجام گردیده است. هدف این آزمایش‌ها ارزیابی عملکرد سیستم یادگیری تقویتی در کارهایی با فضا‌های حالت پیوسته برای چندین مقدار λ بود.

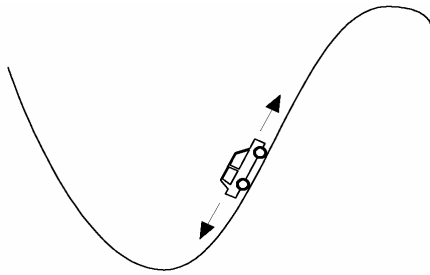
۶-۱- وظایف یادگیری

دو مورد یادگیری مورد آزمایش قرار گرفت. اولین آنها پارک کردن خودرو که توسط سیچوز [20][5] برای ارزیابی مفید بودن دستورالعمل TTD انجام شد و دیگری نسخه‌ای از کار با خودرو در کوهستان [4] که اخیراً برای الگوریتم‌های یادگیری تقویتی با تابع تقریب، مشهور شده است. [15][7]

۶-۱-۱- پارک کردن خودرو

پارک کردن خودرو در شکل شماره (۲) تشریح شده است. یک مستطیل نشانگر خودرو است که در ابتدا در جایی داخل محدوده مرزبندی شده به نام محوطه رانندگی قرار

۶-۱-۲- کار با خودرو در کوهستان



شکل ۳: مسئله خودرو در کوهستان

سه عمل ممکن وجود دارد، "رانندگی به جلو"، "رانندگی به عقب" و "رانندگی به صورت دنده خلاص". اثرات آنها طبق روال شبیه‌سازی در ضمیمه «ب» توصیف شده است.

۶-۲- طرح تجربی و نتایج آن

برای هر دو کار یک سری آزمایش‌ها با مقادیر λ مختلف اجرا گردیده است. پارامترهای مورد استفاده برای این آزمایش‌ها در جدول (۲) نشان داده شده است.

این کار در شکل (۳) تشریح شده است. عامل ملزم است که خودرو را تا بالای یک تپه با شیب تند براند. موتور برای آنکه مستقیماً از نقطه توقف شتاب گرفته و شیب را طی کند بسیار ضعیف است. برای رسیدن به هدف، خودرو می‌بایست ابتدا از آن نقطه دور شود. آموزش در این کار شامل یک سری امتحانات می‌شود که هر یک از حالت غیر هدف اولیه شروع می‌شود و تا رسیدن به هدف ادامه می‌یابد.

وضعیت خودرو در کوهستان با دو متغیر نمایش داده می‌شود، وضعیت افقی خودرو یعنی x و سرعت آن یعنی v . برد مجاز آنها به ترتیب عبارتند از:

$$-0.2 \leq v \leq 0.2, -0.1 \leq x \leq 0.1$$

هر وضعیتی با $x > 0.5$ وضعیت هدف است. وضعیت ابتدایی خودرو ثابت بوده و به صورت $v = 0.0$ و $x = -0.5$ ارائه می‌گردد.

جدول ۲: پارامترهای مورد استفاده در آزمایشات

پارک کردن اتومبیل					خودرو در کوهستان				
λ	m	γ	β	T	λ	m	γ	β	T
0			0.5		0			0.5	
0.3			0.5		0.3			0.5	
0.5	35	0.975	0.5	0.0168	0.5	26	0.982	0.5	0.1856
0.7			0.25		0.7			0.25	
0.9			0.25		0.9			0.1	

قطعیت $1.5/25 = 0.06$ برای x ، $4.0/25 = 0.16$ برای v محقق می‌نماید.

نتایج هر آزمایش بطور متوسط ۲۵ بار اجرا شده که تنها در مقادیر اولیه تولید کننده عدد تصادفی متفاوت بود. هر دور برای ۲۵۰ امتحان در مورد پارک کردن خودرو و ۱۵۰ امتحان برای راندن خودرو در کوهستان ادامه داده شده است. برای کار اولی، نتایج به صورت قطعه‌هایی از تقویت میانگین در هر گام زمانی در ۵ امتحان متوالی قبلی ارائه می‌شود. برای کار بعدی، نتایج به صورت قطعه‌های زمان امتحان میانگین (تعداد

برای کار پارک کردن خودرو از ۶ پوشش CMAC استفاده شد که هر یک شامل $5 \times 5 \times 5$ پوشش بود. در فضای وضعیتی تعیین شده بوسیله $-0.5 \leq x \leq 7.5$ و $0.0 \leq y \leq 12.0$ و $0.9\pi \leq \theta \leq 1.6\pi$. قطعیت این نمودار $8.0/30 = 0.267$ برای x ، $12.0/30 = 0.4$ برای y و $0.7\pi/30 = 0.023\pi$ برای θ است، برای کار خودرو در کوهستان، ۵ تا پوشش استفاده شده که هر یک شامل 5×5 پوشش روی فضای حالت تعیین شده بوسیله $-0.1 \leq x \leq 0.5, -2.0 \leq v \leq 2.0$ می‌باشد. این امرز

گام‌ها در هر امتحان) در مورد پنج امتحان متوالی قبلی نسبت به تعداد امتحان‌ها ارائه می‌گردد.

شکل (۵) منحنی‌های یادگیری حاصله را نشان می‌دهد. لذا می‌توان ملاحظه کرد که:

- برای هر دو کار، ترکیب یادگیری Q بر مبنای TTD و $CMAC$ ، قادر به یادگیری موفقیت‌آمیز بود.
- استفاده از $\lambda > 0$ به ما این اجازه را داد تا شتاب قابل ملاحظه‌ای در یادگیری بدست آوریم.
- مقادیر متوسط λ بهترین عملکرد را محقق نمود.
- λ افزایش یافته باعث ایجاد ضرورت کاهش نرخ یادگیری برای جلوگیری از همگرایی زودرس به سیاست‌های زیر بهینه گردید.

تاثیرات استفاده از مقادیر m متفاوت برای λ ‌های با مقدار نسبتاً بزرگ و ثابت برای مثال پارک کردن خودرو نیز بررسی گردیده است. تقریباً بهترین مقدار $\lambda = 0.9$ و کوچکترین مقدار تست شده برای $m = 5$ است که تشخیص داده شده که این مقدار نسبتاً کوچک و قابل قبول است.

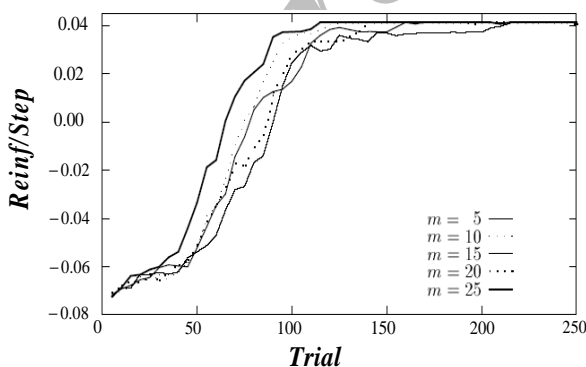
۷- نتیجه‌گیری

این مقاله عملکرد موفقیت‌آمیز ترکیب دستورالعمل TTD و تقریب زنده تابع $CMAC$ را به نمایش گذارده است. همچنین تاثیرات تغییر پارامتر m نیز تحت شرایط خاص برای مثال اول مورد مطالعه قرار گرفت. مورد مفید بودن TTD قبلاً بوسیله شماری از مطالعات تجربی نشان داده شده است. [5][6] به طور مشابه، از $CMAC$ نیز در سیستم‌های یادگیری تقویتی قبلاً توسط تعدادی از مؤلفین بهره گرفته شده است. [15][17] به هر حال کار قبلی روی TTD بطور انحصاری محدود به ارائه یک تابع جدول جستجو بود و کار قبلی بر روی کاربرد $CMAC$ به یادگیری تقویتی متمرکز شده بر ترکیب آن با ساده‌ترین نسخه

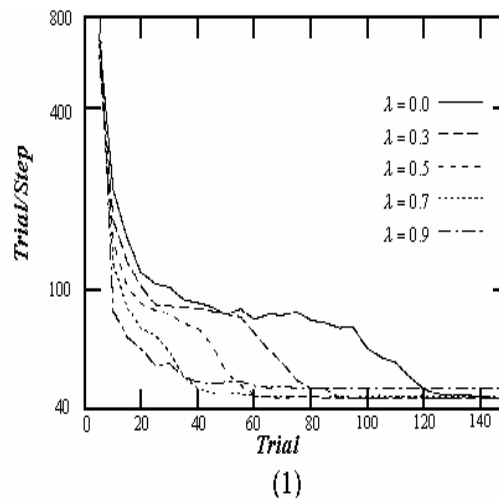
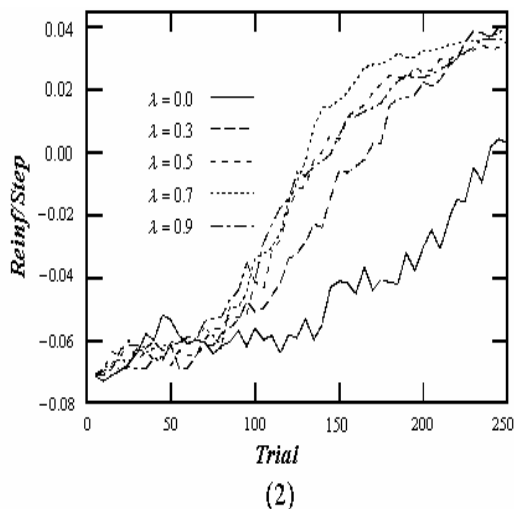
$TD(0)$ از الگوریتم‌های RL و یا استفاده از آثار وجود شرایط پیاده‌سازی $TD(\lambda)$.

در مورد اولی نمی‌توانیم از پتانسیل شتاب یادگیری که با $\lambda > 0$ ممکن می‌شود بهره گرفت. در مورد بعدی، مزایای کارآیی محاسباتی $CMAC$ که وابسته به جمع‌بندی یا تنظیم شمار بسیار کوچکی از وزن‌ها در حین هر عمل که عملیات را ذخیره یا بروز می‌کند است (کسر کوچکی از تعداد کل وزن‌ها)، بطور کامل بوسیله بار تنظیم تمام وزن‌ها از بین می‌رود و علاوه بر آن آثار وجود شرایط برای همه وزن‌ها در خلال هر عمل بروزآوری. دستورالعمل TTD به ما اجازه می‌دهد که $TD(\lambda)$ را برای λ عمومی با نگاهداشتن هزینه‌های محاسباتی پایین $CMAC$ استفاده کنیم.

نتایج تجربی ارائه شده نشان می‌دهد که الگوریتم یادگیری Q بر مبنای TTD با استفاده از $CMAC$ برای نمایش تابع قادر به یادگیری بطور خستگی‌ناپذیر در دو کار با وضعیت مداوم بود. بطور معمول، دستورالعمل TTD به ما اجازه می‌داد که شتاب یادگیری قابل ملاحظه‌ای تحت $TD(0)$ بدون هزینه‌های اضافی چشمگیری بدست آوریم. با ترکیب TTD و $CMAC$ ، این کار، بطور موثری یادگیری سریع، کارآیی محاسباتی و توانمندی‌های تعمیم را ترکیب نمود. این یک گام کوچک است به سوی سنجش یادگیری تقویتی در قلمروهای بزرگ دنیای واقعی.



شکل ۴: مسئله پارک خودرو برای مطالعه و بررسی تاثیرات m با ازای λ ‌های با مقدار نسبتاً بزرگ



شکل ۵: منحنی‌های یادگیری، (۱) مسئله پارک کردن خودرو، (۲) مسئله خودرو در کوهستان

ضمائم:

الف) جزئیات مسئله پارک کردن اتومبیل

برای شبیه‌سازی حرکت اتومبیل در هر گام زمانی از معادلات زیر استفاده شده است:

1. if $r \neq 0$ then

$$(a) \theta(t + \tau) := \theta(t) + \tau \frac{v}{r};$$

$$(b) x(t + \tau) := x(t) - r \sin \theta(t) + r \sin \theta(t + \tau);$$

$$(c) y(t + \tau) := y(t) + r \cos \theta(t) - r \sin \theta(t + \tau);$$

2. if $r = 0$ then

$$(a) \theta(t + \tau) := \theta(t);$$

$$(b) x(t + \tau) := x(t) + \tau v \cos \theta(t);$$

$$(c) y(t + \tau) := y(t) + \tau v \sin \theta(t);$$

که در اینجا r زاویه چرخش، v سرعت اتومبیل، و τ گام زمانی شبیه‌سازی است. در شبیه‌سازی عملی $r = -5$ متر برای عمل گردش به چپ و $r = 5$ متر برای عمل گردش به راست و $r = 0$ برای راه مستقیم در نظر گرفته

شده است. سرعت ثابت و برابر یک متر بر ثانیه و گام‌های شبیه‌سازی $\tau = 0.5$ ثانیه در نظر گرفته شده است.

ب) جزئیات مسئله اتومبیل کوهستان

برای شبیه‌سازی حرکت اتومبیل در کوهستان از معادلات زیر استفاده شده است:

$$1. \text{ if } x_t < 0.0 \text{ then } q := 2x_t + 1$$

$$\text{else } q := \frac{1}{(\sqrt{5x_t^2 + 1})^3};$$

$$2. a_t := \frac{F_t}{\sqrt{q^2 + 1}} - g \frac{q}{q^2 + 1};$$

$$3. x_{t+\tau} := \text{bound}(x_t + \tau v_t + 0.5\tau^2 a_t);$$

$$4. v_{t+\tau} := \text{bound}(v_t + \tau a_t);$$

که $g = 9.81$ و F_t نیروی وارد شده در زمان t که مقدار آن برای عمل رو به جلو 4.0 و برای عمل رو به عقب -

to Back propagation”, Proceedings of the IEEE, 78:1561-1567, 1990

[10] D. Mitchie and R. A. Chambers, “BOXES: An Experiment in Adaptive Control”, Machine Intelligence, 2:137-152, 1968

[11] A. W. Moore, “Efficient Memory-Based Learning for Robot Control”, PHD Thesis, University of Cambridge Computer Laboratory, 1990

[12] S. P. Singh and R. C. Yee, “Technical Note: An Upper Bound on the Loss from Approximate optimal Value Functions”, Machine Learning, 16:227-233, 1994

[13] R.S. Sutton, “Temporal Credit Assignment in Reinforcement Learning”, PHD Thesis, Department of Computer and Information Science, University of Massachusetts, 1984

[14] R. S. Sutton, “Learning to Predict by the Methods of Temporal Differences”, Machine Learning, 3:9-44,1988

[15] R. S. Sutton, “Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding”, To Appear in Advances in Neural Information Processing Systems 8, 1995

[16] S. B. Thurn and A. Schwartz, “Issues in Using Function Approximation for Reinforcement Learning”, In Proceedings of 4th Connectionist Model Summer School, Lawrence Erlbaum, 1993

[17] C. J. C. H. Watkins, “Learning From Delayed Rewards”, PHD thesis, King’s College, Cambridge, 1989

[18] J. Abdi, C. Lux, A. K. Sedigh and A. Famil Khalili, “Truncated Temporal Difference Learning with Function Approximation” 11th International Conference on Electrical Engineering, ICEE’03, Shiraz, Iran, p.p. 283-290, 2003

[19] Tesauro, G., “Practical issues in temporal difference learning”, Machine Learning, Vol. 8, p.p. 257-277, 1992

[20] P. Cichosz, “Truncated Temporal Differences and Sequential Replay: Comparison, Integration, and Experiments”, In Proceedings of the 13th International Conference on Machine Learning, 1996

4.0 و برای عمل سر خوردن مقدار آن برابر صفر است. گام زمانی شبیه سازی 0.03 ثانیه و رنج قابل تغییر $-1.0 \leq x \leq 1.0$ و $-2.0 \leq v \leq 2.0$ در نظر گرفته شده است.

مراجع

[1] J. S. Albus, “Brain, Behavior, and Robotics”. BYTE Books, 1981

[2] L. C. Barid, “Residual Algorithms: Reinforcement Learning with Function Approximation”, In Proceedings of the 12th International Conference on Machine Learning (ML-95), Morgan Kaufmann, 1995

[3] A. G. Barto, R. S. Sutton, and C. W. Anderson, “Neurolike Adaptive Elements that can also Difficult Learning Control Problems”, IEEE Transaction on system, Man, and Cybernetics, 13:835-846, 1983

[4] J. Boyan, and A. W. Moore, “Generalization in Reinforcement Learning: Safely Approximation the Value Function”, In Advances in Neural Information Processing Systems, 7 Morgan Kaufmann, 1995

[5] P. Cichosz and J. J. Mulawka, “Fast and Efficient Reinforcement Learning with Truncated Temporal Differences”, In Proceedings of the 12th International Conference on Machine Learning (ML-95), Morgan Kaufmann, 1995

[6] P.Cichosz, “Truncating Temporal Differences: On the Efficient Implementation of TD(λ) for Reinforcement Learning”, Journal of Artificial Intelligence Research, 2:287-318, 1995

[7] G. J. Gordon, “Stable Function Approximation in Dynamic Programming”, In Proceedings of the 12th International Conference on Machine Learning (ML-95), Morgan Kaufmann, 1995

[8] Long-Ji Lin, “Reinforcement Learning for Robots Using Neural Network”, PHD Thesis, School of Computer Science, Carnegie-Mellon University, January 1993

[9] W. T. Miller, F.H. Glanz, and L. G. Kraft, “CMAC: An associative Neural Network Alternative

