روش تفاضل‌های زمانی

سینا ایروانیان

بهار ۱۳۸۸

فهرست مطالب

۱ مقدّمه **۱**

۲ کاربرد روش تفاضل‌های زمانی در مسئله‌ی پیش‌بینی **۱**
۱.۲ پیش‌بینی تک‌مرحله‌ای و پیش‌بینی چندمرحله‌ای ۲
۲.۲ مشکلات محاسباتی ۳
۳.۲ خانواده‌ی روش‌های یادگیری $\mathrm{TD}(\lambda)$ ۶
۴.۲ مثالی از یک بازی ۷
۵.۲ مثال قدم برداشتن تصادفی ۹

۳ کاربرد در یادگیری تقویتی **۱۰**
۱.۳ مقدّمه‌ای بر یادگیری تقویتی ۱۱
۲.۳ استفاده از روش تفاضل‌های زمانی در یادگیری تقویتی ۱۳
۳.۳ مسئله‌ی تصمیم‌گیری ۱۶
۴.۳ مثال‌ها ۱۸
۱.۴.۳ قدم برداشتن تصادفی ۱۸
۲.۴.۳ جهان مشبّک ۱۹
۳.۴.۳ جهان مشبّک طوفانی ۲۰
۴.۴.۳ راه‌رفتن کنار درّه ۲۳

۴ نرم‌افزار طرّاحی شده **۲۵**
۱.۴ فایل‌های برنامه ۳۰

لیست تصاویر

۱ مثالی از یک وضعیت بازی ۹
۲ مسئله‌ی قدم برداشتن تصادفی ۹
۳ میانگین خطا در مسئله‌ی پیش‌بینی قدم‌برداشتن تصادفی به‌ازای λهای مختلف . ۱۱
۴ میانگین خطا در مسئله‌ی پیش‌بینی قدم‌برداشتن تصادفی به‌ازای αهای مختلف . ۱۲
۵ حلّ مسئله‌ی قدم برداشتن تصادفی با ۵ گره درونی با استفاده از یادگیری تقویتی . ۱۹
۶ حلّ مسئله‌ی قدم برداشتن تصادفی با ۴۹ گره درونی با استفاده از یادگیری تقویتی ۲۰
۷ مسئله‌ی جهان مشبّک ۲۱
۸ حلّ مسئله‌ی جهان مشبّک ۷ × ۱۰ . . . . . ۲۱
۹ حلّ مسئله‌ی جهان مشبّک ۲۰ × ۲۰، با حرکت شاه ۲۲
۱۰ حلّ مسئله‌ی جهان مشبّک طوفانی ۲۲
۱۱ حلّ مسئله‌ی جهان مشبّک طوفانی، با حرکت شاه ۲۳
۱۲ حلّ مسئله‌ی راه رفتن کنار درّه با استفاده از الگوریتم SARSA ۲۴
۱۳ حلّ مسئله‌ی راه رفتن کنار درّه با استفاده از الگوریتم Q-Learning ۲۴
۱۴ رابط گرافیکی اصلی برنامه ۲۶
۱۵ اجرای نمونه‌ای از مسئله‌ی پیش‌بینی قدم‌برداشتن تصادفی ۲۷
۱۶ اجرای نمونه‌ای از مسئله‌ی قدم‌برداشتن تصادفی به کمک یادگیری تقویتی . . . ۲۸
۱۷ اجرای یکی از مسائل جهان مشبّک ۳۰

چکیده

در این نوشتار به بررسی روش تفاضل‌های زمانی، و کاربرد آن در مسئله‌ی پیش‌بینی و یادگیری تقویتی خواهیم پرداخت. در انتها نیز نرم‌افزاری را که الگوریتم‌های مرتبط با آن را پیاده‌سازی می‌کند ارائه خواهیم داد. روش تفاضل‌های زمانی در پیش‌بینی مقادیر آینده‌ی یک دنباله از مشاهدات کاربرد دارد. همچنین یکی از پرکاربردترین و اصلی‌ترین ایده‌های به کار گرفته شده در یادگیری تقویتی است. روش تفاضل‌های زمانی یک روش یادگیری است؛ که نه در دسته‌بندی یادگیری با نظارت قرار می‌گیرد و نه یادگیری بی‌نظارت. صورت بندی خاصّی از این روش، عملکردی شبیه به یادگیری بانظارت دارد، امّا با سرعت بسیار بالاتر و حافظه‌ی مصرفی بسیار کم‌تر. در حالت کلّی، کیفیت یادگیری این روش متفاوت با روش یادگیری بانظارت است. استفاده از روش تفاضل‌های زمانی در ایده‌ی یادگیری تقویتی، منجر به الگوریتم‌های یادگیری تقویتی و الگوریتم‌های تصمیم‌گیری SARSA و Q-Learning می‌شود. مثال‌های متنوّع ارائه شده در این نوشتار همگی به زبان Matlab پیاده‌سازی شده‌اند. در بخشی جداگانه به راهنمای استفاده از نرم‌افزار طرّاحی شده پرداخته شده است.

۱ مقدّمه

روش تفاضل‌های زمانی[۱] یک روش یادگیری است. آن‌چه یاد می‌گیرد، مقادیری است که یک کمیّت یا سیگنال می‌تواند در آینده اختیار کند. به‌همین دلیل آن را یک روش پیش‌بینی نیز می‌دانند. روش‌های یادگیری به سه دسته تقسیم می‌شوند: *یادگیری بانظارت*[۲]، *یادگیری بی‌نظارت*[۳]، و *یادگیری تقویتی*[۴]. در یادگیری با نظارت عامل با یک سری ورودی و پاسخ صحیح از قبل تعیین شده آموزش می‌بیند. به واحدی که ورودی را به همراه پاسخ صحیح فراهم می‌کند، می‌گویند *ناظر*، و به این مجموعه‌ی ورودی‌ها و پاسخ‌های صحیح می‌گویند *مجموعه‌ی آموزشی*[۵]. در نهایت عامل پس از یادگیری مجموعه‌ی آموزشی باید بتواند پاسخ ورودی‌های جدیدی که تابحال تجربه نکرده است را پیش‌بینی کند. در یاگیری بی‌نظارت واحدی به‌عنوان ناظر وجود ندارد؛ و عامل سعی می‌کند با توجه به الگوهای مشاهده شده در داده‌های ورودی آن‌ها را کلاسه‌بندی یا دسته‌بندی کند. در یادگیری تقویتی عامل از طریق تعامل با محیط، و سعی و خطا تلاش می‌کند عمل یادگیری را انجام دهد. در این مسئله، عامل باید هدف و نحوه‌ی تعامل با محیط را بداند؛ امّا لازم نیست که مدل تغییر و تحوّل محیط را در اثر گذشت زمان بداند. پس از هر کنشی که عامل انجام می‌دهد، محیط از یک وضعیت به وضعیت دیگر می‌رود و با توجّه به این تغییر وضعیت عامل باید میزان سودمندی کنشی که انجام داده را محاسبه کند که در ادبیات یادگیری تقویتی به آن *پاداش*[۶] و در ادبیات تئوری کنترل به آن *سیگنال تقویتی*[۷] می‌گویند.

اگر قرار باشد یک ایده را به عنوان ایده‌ی مرکزی و بدیع یادگیری تقویتی معرفی کنیم، این ایده بدون شک یادگیری تفاضل زمانی خواهد بود [۱]. زین پس در این نوشتار، روش تفاضل‌های زمانی را به اختصار TD خواهیم خواند. در بخش ۲، ابتدا به کاربرد روش تفاضل‌های زمانی در مسئله‌ی پیش‌بینی خواهیم پرداخت. در بخش ۳ مقدّمه‌ای بر یادگیری تقویتی ارائه خواهد شد و در آن‌جا خواهیم گفت که چگونه می‌توان مسئله‌ی یادگیری تقویتی را به‌صورت حالت خاصّی از مسئله‌ی پیش‌بینی به کمک روش تفاضل‌های زمانی در نظر گرفت و آن را حل کرد. بخش ۴ نیز به معرّفی نرم‌افزار طرّاحی شده و طرز کار آن می‌پردازد.

۲ کاربرد روش تفاضل‌های زمانی در مسئله‌ی پیش‌بینی

در این بخش به کاربرد روش تفاضل‌های زمانی در مسئله‌ی پیش‌بینی خواهیم پرداخت. ابتدا مسئله‌ی پیش‌بینی و انواع آن را معرّفی می‌کنیم و رهیافت یادگیری با نظارت را برای حلّ این گونه مسائل ارائه خواهیم داد. سپس نشان می‌دهیم که چگونه می‌توان این راه حل را از نظر زمان اجرایی و حافظه‌ی

[۱] Temporal Differences
[۲] Supervised Learning
[۳] Unsupervised Learning
[۴] Reinforcement Learning
[۵] Training-Set or Training-Data
[۶] reward
[۷] reinforcement signal

مصرفی بهبود داد؛ و در نهایت کلاسی از روش‌های تفاضل‌های زمانی را معرّفی خواهیم کرد که کیفیت یادگیری آن با روش یادگیری با نظارت متفاوت است.

۱.۲ پیش‌بینی تک‌مرحله‌ای و پیش‌بینی چندمرحله‌ای

می‌توان مسئله‌ی پیش‌بینی را به دو دسته‌ی پیش‌بینی *تک‌مرحله‌ای*[۸] و پیش‌بینی *چندمرحله‌ای*[۹] تقسیم کرد. در پیش‌بینی تک‌مرحله‌ای، صحت پیش‌بینی انجام شده بلافاصله مشخّص می‌شود. در حالی که در پیش‌بینی چندمرحله‌ای، صحت پیش‌بینی انجام شده تنها پس از گذشت چند مرحله از انجام پیش‌بینی مشخص می‌شود؛ البتّه که در هر مرحله نیز یک سری اطلاعات جزئی و ناقص در رابطه با صحت پیش‌بینی انجام شده مشخص می‌شود. به عنوان مثال فرض کنید قصد داریم از طریق مشاهده‌ی وضع هوای دوشنبه، پیش‌بینی کنیم که در جمعه باران می‌بارد یا نه. این مسئله‌ی پیش‌بینی وضع هوا، نمونه‌ای از مسئله‌ی پیش‌بینی چند مرحله‌ای است؛ چرا که شواهد ناقص مربوط به صحت پیش‌بینی وضع هوای جمعه، در روزهای سه‌شنبه، چهارشنبه و پنج‌شنبه مشخّص می‌شوند؛ اما ارزیابی نهایی صحت پیش‌بینی انجام شده، تنها در روز جمعه امکان پذیر است. اما اگر بخواهیم با مشاهده‌ی وضع هوای هر روز، تنها وضع هوای روز بعد را پیش‌بینی کنیم (یعنی در دوشنبه، وضع هوای سه‌شنبه را پیش‌بینی کنیم؛ در سه‌شنبه وضع هوای چهارشنبه را پیش‌بینی کنیم؛ و به همین ترتیب) آن‌گاه با یک مسئله‌ی پیش‌بینی تک‌مرحله‌ای روبرو هستیم؛ به شرطی که بین زمان انجام پیش‌بینی، و تأیید یا تکذیب آن در روز بعد، هیچ مشاهده‌ی دیگری انجام نشده باشد.

در این نوشتار، ما تنها به مسئله‌ی پیش‌بینی چندمرحله‌ای می‌پردازیم. در پیش‌بینی تک‌مرحله‌ای داده‌ها به طور طبیعی به صورت زوج مشاهده-نتیجه هستند؛ که این مسئله یک نمونه از مسئله‌ی یادگیری با نظارت است. روش تفاضل‌های زمانی، تنها در مسئله‌ی پیش‌بینی چند مرحله‌ای از روش‌های یادگیری با نظارت بهتر عمل می‌کنند؛ اما مسائل دنیای واقعی، بیشتر از جنس پیش‌بینی چند مرحله‌ای هستند [۲]. مثلاً پیش‌بینی وضع اقتصادی در سال آینده، بلافاصله قابل تأیید یا تکذیب نیست؛ بلکه مرحله به مرحله با مشاهده‌ی وضع اقتصادی در طول سال قابل انجام است. تخمین احتمال سرنوشت یک انتخابات، با هر نظر سنجی بهبود می‌یابد؛ و تخمین احتمال نتیجه‌ی یک بازی شطرنج، پس از هر حرکت روشن‌تر می‌شود.

در واقع، بسیاری از مسائلی که به طور کلاسیک، در دسته بندی پیش‌بینی تک‌مرحله‌ای (زوج الگو-نتیجه در یادگیری بانظارت) قرار می‌گیرند، ماهیتاً از جنس پیش‌بینی چند مرحله‌ای هستند. به طور کلاسیک، با مسائل ادراکی، مثل تشخیص صدا یا تصویر؛ به وسیله‌ی روش‌های یادگیری بانظارت برخورد می‌شود؛ که ورودی آن یک مجموعه‌ی یادگیری متشکل از نمونه‌های مجزا که به درستی کلاسه‌بندی شده‌اند می‌باشد. در حالی که، وقتی انسان‌ها صدایی را می‌شنوند، یا چیزی را می‌بینند، در طول زمان جریانی از سیگنال‌های ورودی را دریافت می‌کنند، و به مرور فرض خود مبنی بر این که چه می‌شنوند یا چه می‌بینند را بهبود می‌بخشند. انسان‌ها با یک مسئله‌ی تک‌مرحله‌ای تشخیص الگو به صورت آن‌چه در یادگیری با نظارت انجام می‌شود روبرو نیستند؛ بلکه با یک سری

[۸] single-step

[۹] multi-step

الگوهای مرتبط که هریک اطلاعاتی در رابطه با یک دسته‌بندی واحد ارائه می‌دهند، مواجه هستند.

۲.۲ مشکلات محاسباتی

ورودی مسئله‌ی پیش‌بینی چند مرحله‌ای، به صورت دنباله‌ی مشاهدات-نتیجه، به فرم

$$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m, z$$

است که در آن $\mathbf{x}_t$ بردار مشاهده در زمان t در دنباله‌ی فوق، و z نتیجه‌ی دنباله است. درایه‌های $\mathbf{x}_t$ اعداد حقیقی هستند (برداری از اندازه‌گیری‌ها یا خصوصیات محیط) و z نیز یک اسکالر حقیقی است.

به ازای هر دنباله‌ی مشاهدات-نتیجه، یادگیرنده یک دنباله‌ی پیش‌بینی متناظر تولید می‌کند

$$P_1, P_2, \ldots, P_m$$

که در آن P_t ها پیش‌بینی‌ای از مقدار z هستند. به طور کلی P_t می‌تواند تابعی از $\mathbf{x}_t$ به تنهایی باشد، یا تابعی از تمام مشاهدات از $\mathbf{x}_1$ تا $\mathbf{x}_t$ باشد. پیش‌بینی‌ها همچنین تابعی از برداری از پارامترهای تغییرپذیر یا، بردار وزن‌ها هستند، که آن را با $\mathbf{w}$ نشان می‌دهیم؛ و وابستگی تابعی P_t به $\mathbf{x}_t$ و $\mathbf{w}$ را نیز با $P(\mathbf{x}_t, \mathbf{w})$ نشان می‌دهیم.

تمامی روش‌های یادگیری، به صورت قوانینی برای تنظیم $\mathbf{w}$ بیان می‌شوند. در این‌جا ابتدا فرض می‌کنیم که $\mathbf{w}$ تنها یک‌بار آن هم در انتهای دنباله‌ی مشاهدات به‌هنگام می‌شود؛ و در طول دنباله $\mathbf{w}$ هیچ تغییری نمی‌کند. برای هر مشاهده، یک افزایش برای مقدار $\mathbf{w}$ در نظر گرفته می‌شود، که آن را با $\Delta\mathbf{w}_t$ نشان می‌دهیم؛ اما تنها هنگامی که کلّ دنباله‌ی مشاهدات به پایان رسید، مجموع این افزایش‌ها به $\mathbf{w}$ اعمال می‌شود:

$$\mathbf{w} \leftarrow \mathbf{w} + \sum_{t=1}^{m} \Delta\mathbf{w}_t \tag{1}$$

یادگیری با نظارت به هر دنباله‌ی مشاهدات-نتیجه، به صورت دنباله‌ای از زوج‌های مشاهده-نتیجه برخورد می‌کند، یعنی به صورت دنباله‌ی

$$(\mathbf{x}_1, z), (\mathbf{x}_2, z), \ldots, (\mathbf{x}_m, z)$$

در این صورت میزان افزایش وزن در زمان t، متناسب خواهد بود با خطای پیش‌بینی انجام شده در زمان t (یعنی تفاضل P_t و z)، و متناسب خواهد بود با این که تغییر در $\mathbf{w}$ چه تأثیری در P_t خواهد داشت. شمای کلّی روند تغییر وزن در یادگیری با نظارت به صورت زیر است:

$$\Delta\mathbf{w}_t = \alpha(z - P_t)\nabla_w P_t \tag{2}$$

که در آن α یک پارامتر مثبت است، که نرخ یادگیری را مشخّص می‌کند، و گرادیان $\nabla_w P_t$، بردار مشتقّات جزئی P_t نسبت به درایه‌های $\mathbf{w}$ است.

به عنوان مثال حالت خاصّی را در نظر بگیرید که در آن P_t تابع خطّی‌ای از $\mathbf{x}_t$ و $\mathbf{w}$ است، یعنی

$$P_t = \mathbf{w}^T\mathbf{x}_t = \sum_i \mathbf{w}(i)\mathbf{x}(i)$$

با این فرض داریم، $\nabla_w P_t = \mathbf{x}_t$، که با جایگذاری، رابطه‌ی (۲)، به قانون معروف Widrow-Hoff برای بروزرسانی وزن‌ها کاهش می‌یابد [۳]:

$$\Delta\mathbf{w}_t = \alpha(z - \mathbf{w}^T\mathbf{x}_t)\mathbf{x}_t \qquad (۳)$$

به این روش یادگیری خطّی، «قانون دلتا»، و فیلتر LMS نیز می‌گویند. روش‌های دیگری برای محاسبه‌ی پیش‌بینی به کمک بردار وزن‌ها نیز وجود دارند، که در آن پیش‌بینی تابع خطی از بردار وزن‌ها و بردار مشاهده نمی‌باشد، مانند روش backpropagation [۴]. در این روش محاسبه‌ی گرادیان $\nabla_w P_t$ نیز پیچیده‌تر است.

در هر صورت، محاسبه‌ی $\Delta\mathbf{w}_t$ در (۲) به طور کامل وابسته است به مقدار z، و مقدار z نیز تا پایان دنباله‌ی مشاهدات مشخص نمی‌شود. بنابراین تمامی مشاهدات و پیش‌بینی‌های انجام شده در طول دنباله، باید جایی ذخیره شوند، تا در انتها بتوانیم $\Delta\mathbf{w}_t$ ها را محاسبه کنیم. به بیان دیگر رابطه‌ی (۲) را نمی‌توان به صورت افزایشی[۱۰] محاسبه کرد.

با این حال، یک روش تفاضل زمانی وجود دارد، که نتیجه‌اش درست مانند نتیجه‌ی (۲) است، اما به صورت افزایشی قابل محاسبه است. ایده‌ی آن، نمایش خطای $z - P_t$ به صورت مجموع تغییرات در پیش‌بینی‌هاست، یعنی به صورت زیر:

$$z - P_t = \sum_{k=t}^{m}(P_{k+۱} - P_k) \quad \text{و} \quad P_{m+۱} \stackrel{\text{تعریف}}{=} z$$

که با استفاده از آن، و ترکیب با (۱) و (۲)، روابط زیر بدست می‌آیند:

$$\begin{aligned}\mathbf{w} &\leftarrow \mathbf{w} + \sum_{t=۱}^{m}\alpha(z - P_t)\nabla_w P_t \\ &= \mathbf{w} + \sum_{t=۱}^{m}\alpha\sum_{k=t}^{m}(P_{k+۱} - P_k)\nabla_w P_t\end{aligned}$$

عبارت فوق به صورت افزایشی قابل محاسبه نیست. با تکنیکی شبیه به جابجایی انتگرال‌ها و تبدیل حدود در انتگرال‌های دو گانه، جای دو جمع در عبارت فوق را عوض می‌کنیم:

$$\begin{aligned}\mathbf{w} &\leftarrow \mathbf{w} + \sum_{k=۱}^{m}\alpha\sum_{t=۱}^{k}(P_{k+۱} - P_k)\nabla_w P_t \\ &= \mathbf{w} + \sum_{t=۱}^{m}\alpha(P_{t+۱} - P_t)\sum_{k=۱}^{t}\nabla_w P_k\end{aligned}$$

[۱۰] incremental

و در نتیجه رابطه‌ی به روز رسانی وزن‌ها در هر مرحله، که قابل استفاده در (۱) نیز باشد، به صورت زیر خواهد بود:

$$\Delta \mathbf{w}_t = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \nabla_w P_k \qquad (۴)$$

رابطه‌ی بالا بر خلاف (۲)، به صورت افزایشی قابل محاسبه است. به این صورت که هنگام مشاهده‌ی $\mathbf{x}_{t+1}$، و انجام تخمین P_{t+1}، تغییر وزن مربوط به مرحله‌ی قبل، یعنی $\Delta \mathbf{w}_t$، با استفاده از رابطه‌ی (۴) محاسبه می‌شود، که در آن P_t پیش‌بینی انجام شده در زمان t، و P_{t+1} پیش‌بینی انجام شده در زمان جاری است، که وابسته به مقدار بردار وزن و بردار مشاهده است، یعنی $P_{t+1} = P(\mathbf{w}_t, \mathbf{x}_{t+1})$. همچنین جمع گرادیان‌ها در (۴) نیز به طور افزایشی قابل محاسبه است، و هزینه‌ی محاسبه‌ی آن در طول دنباله‌ی مشاهدات پخش می‌شود. پس از هر مرحله بردار وزن به صورت $\mathbf{w}_{t+1} = \mathbf{w}_t + \Delta \mathbf{w}_t$ بروزرسانی می‌شود تا در مرحله‌ی بعد مورد استفاده قرار بگیرد.

اگر بیشترین طول دنباله‌ی مشاهدات را M فرض کنیم، آن‌گاه غالباً، محاسبه‌ی رابطه‌ی (۴)، نیاز به $\frac{1}{M}$ حافظه و سرعت پردازشگر لازمه برای محاسبه‌ی رابطه‌ی (۲) خواهد داشت.

به دلایلی که بعداً توضیح داده خواهد شد، به الگوریتم بروزرسانی وزن‌های (۴)، TD(1) گفته می‌شود، که منظور از TD همان روش تفاضل‌های زمانی است. به خصوص، حالت خاصّی که در آن P_t تابعی خطّی از بردار مشاهده‌ی $\mathbf{x}_t$ و بردار وزن‌ها $\mathbf{w}$ می‌باشد، یعنی $P_t = \mathbf{w}^T \mathbf{x}_t$، برای ما اهمّیت دارد. در این حالت به روش تفاضل‌های زمانی متناظر، *روش تفاضل‌های زمانی خطّی* گفته می‌شود. رابطه‌ی افزایش وزنهای (۴) برای TD(1) خطّی بصورت زیر خواهد بود:

$$\Delta \mathbf{w}_t = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \mathbf{x}_k \qquad\qquad P_t = \mathbf{w}^T \mathbf{x}_t$$

الگوریتم ۱ روش تفاضل‌های زمانی TD(1) خطّی، که به طور افزایشی ارائه شده است، را نشان می‌دهد. گاهی لازم است برای کاهش خطای یادگیری این الگوریتم را چندین بار به ورودی‌های مختلف یا حتّی یکسان اعمال کرد؛ و هر بار مقدار اوّلیه‌ی بردار وزن‌ها را با مقادیر آن در آخرین مرحله‌ی یادگیری مقداردهی اوّلیه کرد.

با مطالبی که تا کنون گفته شد، قضیه‌ی زیر را ثابت کرده‌ایم:

قضیه ۱ *در مسائل پیش‌بینی چند مرحله‌ای، روش TD(1) خطّی در انتهای دنباله‌ی مشاهدات، بردار وزن‌ها را به همان اندازه تغییر می‌دهد که روش Widrow-Hoff بردار وزن‌ها را تغییر می‌دهد.*

بنابراین با این که روش TD(1) یک راهکار افزایشی نسبت به یادگیری با نظارت ارائه می‌دهد، اما بردار وزن‌های حاصل در نهایت تفاوتی با بردار وزن‌های بدست آمده با روش یادگیری با نظارت ندارد. به عبارت دیگر کیفیت یادگیری تغییری نکرده. در بخش بعدی خانواده‌ای از روش‌های تفاضل‌های زمانی را معرّفی می‌کنیم، که تغییر در بردار وزن‌های آن، با تمامی روش‌های یادگیری با نظارت دیگر متفاوت است.

الگوریتم ۱ الگوریتم TD(1) خطّی

ورودی: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m, z$ ▷ دنباله‌ی مشاهدات-نتیجه

خروجی: $\mathbf{w}$ ▷ بردار وزن‌ها

$\mathbf{w} \leftarrow \mathbf{w}_0$ ▷ بردار وزن‌ها را با مقادیر دلخواه مقداردهی اوّلیه کن

$\Delta\mathbf{w} \leftarrow 0$

$P_t \leftarrow \mathbf{w}^T\mathbf{x}_1$ ▷ $\mathbf{x}_1$ مشاهده شد

$\mathbf{S}_t \leftarrow \mathbf{x}_1$ ▷ $\mathbf{S}_t$ همان جمع گرادیان‌هاست

for all $\mathbf{x}_t, t = 2, \ldots, m$ **do** ▷ برای مشاهدات $\mathbf{x}_2$ تا $\mathbf{x}_m$

  $P_{t-1} \leftarrow P_t$

  $P_t \leftarrow \mathbf{w}^T\mathbf{x}_t$

  $\Delta\mathbf{w} \leftarrow \Delta\mathbf{w} + \alpha(P_t - P_{t-1})\mathbf{S}_t$ ▷ میزان افزایش بردار وزن‌ها برای مشاهده‌ی قبلی

  $\mathbf{S}_t \leftarrow \mathbf{S}_t + \mathbf{x}_t$

end for

$\Delta\mathbf{w} \leftarrow \Delta\mathbf{w} + \alpha(z - P_t)\mathbf{S}_t$

$\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w}$ ▷ بروزرسانی بردار وزن‌ها

۳.۲ خانواده‌ی روش‌های یادگیری TD(λ)

بارزترین خصوصیت روش تفاضل‌های زمانی، حساسیت آن‌ها به تغییرات در پیش‌بینی‌های متوالی، به جای خطای کلی بین هر پیش‌بینی و نتیجه‌ی نهایی است. در پاسخ به هر افزایش (کاهش) در پیش‌بینی از P_t به P_{t+1}، افزایش در وزن‌های $\Delta\mathbf{w}_t$ طوری تعیین می‌شود که همه یا برخی از پیش‌بینی‌های مشاهدات گذشته، $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t$، نیز افزایش (کاهش) می‌یابد. مثلاً رابطه‌ی (۴)، حالت خاصّی است که تمامی پیش‌بینی‌های گذشته را به یک میزان تغییر می‌دهد. در این‌جا کلاسی از روش‌های تفاضل‌های زمانی را معرفی می‌کنیم که پیش‌بینی‌های جدیدتر را بیش‌تر از پیش‌بینی‌های گذشته تغییر می‌دهد. به طور خاص، یک روش وزن‌دهی نمایی با تأخّر، را در نظر می‌گیریم که در آن تغییر در پیش‌بینی مشاهداتی که در k مرحله پیش انجام شدند متناسب است با λ^k برای $0 \leq \lambda \leq 1$:

$$\Delta\mathbf{w}_t = \alpha(P_{t+1} - P_t)\sum_{k=1}^{t}\lambda^{t-k}\nabla_w P_k \tag{۵}$$

توجه کنید که به‌ازای $\lambda = 1$، رابطه‌ی فوق معادل خواهد شد با رابطه‌ی (۴)، که پیاده‌سازی روش یادگیری با نظارت به صورت تفاضل‌های زمانی بود. روش جدید TD(λ)، نامیده می‌شود، و متناظراً روش ارائه شده در (۴)، TD(1) نامیده می‌شود.

به جای وزن دهی نمایی در (۵)، می‌توان از روش‌های دیگری نیز، بسته به کاربرد، استفاده کرد؛ اما مزیت وزن‌دهی نمایی این است که آن را می‌توان به صورت افزایشی محاسبه کرد. مثلاً فرض کنید که مقدار جمع در (۵) را برای مرحله‌ی t، با s_t نمایش دهیم، در این صورت مقدار s_{t+1} را

می‌توان به صورت افزایشی محاسبه کرد:

$$\begin{aligned} s_{t+1} &= \sum_{k=1}^{t+1} \lambda^{t+1-k} \nabla_w P_k \\ &= \nabla_w P_{t+1} + \sum_{k=1}^{t} \lambda^{t+1-k} \nabla_w P_k \\ &= \nabla_w P_{t+1} + \lambda\, s_t \end{aligned}$$

به ازای $\lambda < 1$، خانواده‌ی روش‌های تفاضل‌های زمانی، بردار وزن‌ها را به شکلی کاملاً متمایز با تمامی روش‌های یادگیری با نظارت تغییر می‌دهد. به خصوص به ازای $\lambda = 0$، این مطلب مشهودتر است. در TD(0)، میزان افزایش در بردار وزن‌ها متناسب است با تأثیر آن در آخرین پیش‌بینی:

$$\Delta \mathbf{w}_t = \alpha (P_{t+1} - P_t) \nabla_w P_t \tag{6}$$

به شباهت رابطه‌ی فوق با (۲) توجه کنید. تنها تفاوت در این است، که در رابطه‌ی فوق به جای نتیجه‌ی نهایی z، مقدار پیش‌بینی بعدی، P_{t+1}، قرار گرفته است. در واقع هدف از تغییر وزن در (۲) نزدیک شدن پیش‌بینی‌ها به z است، در حالی که در رابطه‌ی فوق هدف از تغییر وزن، نزدیک شدن به P_{t+1} می‌باشد.

الگوریتم ۲، روش TD(λ)را نشان می‌دهد. به ماهیت افزایشی بودن این روش توجه کنید. گاهی لازم است برای کاهش خطای یادگیری این الگوریتم را چندین بار به ورودی‌های مختلف یا حتّی یکسان اعمال کرد؛ و هر بار مقدار اوّلیه‌ی بردار وزن‌ها را با مقادیر آن در آخرین مرحله‌ی یادگیری مقداردهی اوّلیه کرد. الگوریتم TD(λ)خطّی، حالت خاصّ الگوریتم ارائه شده می‌باشد، که در آن $P(\mathbf{w}, \mathbf{x}_t) = \mathbf{w}^T \mathbf{x}_t$ و $\nabla_w P_t = \mathbf{x}_t$.

۴.۲ مثالی از یک بازی

تا این لحظه دیدیم که روش تفاضل‌های زمانی از نظر سرعت اجرا و حافظه‌ی مصرفی نسبت به یادگیری با نظارت برتری دارد؛ و همچنین به‌ازای $\lambda < 1$، کیفیت یادگیری این روش کاملاً متمایز است نسبت به یادگیری با نظارت. اکنون مثالی را مطرح می‌کنیم که در آن تنها روش تفاضل‌های زمانی پاسخ صحیح را ارائه می‌دهد؛ و پاسخی که روش یادگیری با نظارت ارائه می‌دهد تحت شرایط خاص و در دراز مدّت به پاسخ صحیح همگرا می‌شود.

به عنوان مثال یک وضعیت بازی را در نظر بگیرید که عامل تا کنون یادگرفته که این وضعیت بد است؛ چرا که اکثر اوقات به باخت منجر می‌شود، و به ندرت به بُرد منجر شده است. شکل ۱، یک نمونه‌ی ساده‌ی چنین موقعیتی را به صورت یک وضعیت «بَد» نمایش می‌دهد که در ۹۰٪ مواقع به «باخت» منجر شده و تنها در ۱۰٪ باقی مواقع به «بُرد» منتهی شده است. اکنون فرض کنید که در حین بازی به یک وضعیت جدید می‌رسیم (وضعیتی که تا قبل از این تجربه نشده بود)، که این وضعیت جدید به وضعیت «بَد» منجر شده و سپس به «بُرد» ختم می‌شود. این مسیر در شکل ۱،

الگوریتم ۲ الگوریتم $\mathrm{TD}(\lambda)$

ورودی: $\mathbf{x}_۱, \mathbf{x}_۲, \ldots, \mathbf{x}_m, z$ ▷ دنباله‌ی مشاهدات–نتیجه

خروجی: $\mathbf{w}$ ▷ بردار وزن‌ها

$\mathbf{w} \leftarrow \mathbf{w}_۰$ ▷ بردار وزن‌ها را با مقادیر دلخواه مقداردهی اوّلیه کن

$\Delta\mathbf{w} \leftarrow ۰$

$P_t \leftarrow P(\mathbf{w}, \mathbf{x}_۱)$ ▷ $\mathbf{x}_۱$ مشاهده شد

$\mathbf{S} \leftarrow \nabla_w P_t$ ▷ $\mathbf{S}$ همان جمع وزن‌دار گرادیان‌هاست

for all $\mathbf{x}_t, t = ۲, \ldots, m$ **do** ▷ برای مشاهدات $\mathbf{x}_۲$ تا $\mathbf{x}_m$

  $P_{t-۱} \leftarrow P_t$

  $P_t \leftarrow P(\mathbf{w}, \mathbf{x}_t)$

  $\Delta\mathbf{w} \leftarrow \Delta\mathbf{w} + \alpha(P_t - P_{t-۱})\mathbf{S}$ ▷ میزان افزایش بردار وزن‌ها برای مشاهده‌ی قبلی

  $\mathbf{S} \leftarrow \nabla_w P_t + \lambda\mathbf{S}$

end for

$\Delta\mathbf{w} \leftarrow \Delta\mathbf{w} + \alpha(z - P_t)\mathbf{S}$

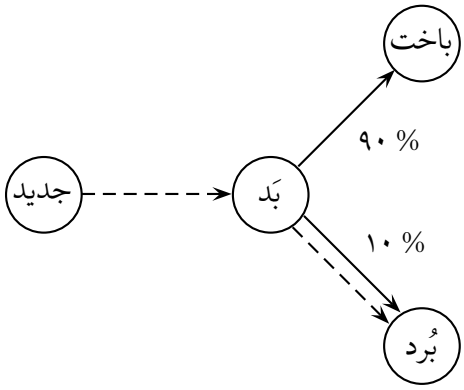
$\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w}$ ▷ بروزرسانی بردار وزن‌ها

با خط‌چین نشان داده شده است. پس از این مشاهده نظرمان نسبت به وضعیت «بد» کمی بهبود می‌یابد، امّا وضعیت «جدید» را چگونه ارزش‌دهی باید کرد؟

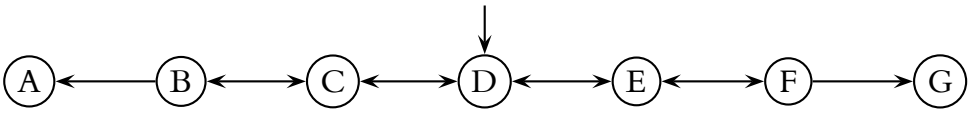
از آن‌جا که در تمامی مشاهداتی که تا کنون انجام شده، وضعیت «جدید» کاملاً به «بُرد» منجر شده؛ یادگیری با نظارت زوج مرتّبی متشکل از وضعیت «جدید» و «بُرد» تشکیل می‌دهد، و نتیجه‌گیری می‌کند که وضعیت جدید به بُرد منجر می‌شود و وضعیت خوبی است. اما یک روش تفاضل‌های زمانی، زوج مرتّبی متشکل از وضعیت «جدید» و وضعیت «بَد» که بلافاصله پس از آن مشاهده شد تشکیل می‌دهد؛ و نتیجه می‌گیرد که وضعیت «جدید» خود یک وضعیت بد است و اکثر اوقات به باخت منجر خواهد شد. با فرض این که تا قبل از این مشاهده ارزش داده شده به وضعیت «بد» صحیح بوده باشد، ارزش‌دهی‌ای که روش تفاضل‌های زمانی انجام می‌دهد درست‌تر از ارزش‌دهی‌ای است که روش یادگیری با نظارت انجام می‌دهد. البتّه در دراز مدّت به شرط آن که وضعیت «جدید» به تعداد کافی زیاد مشاهده شود، هر دو روش به یکدیگر همگرا خواهند شد؛ امّا روش تفاضل‌های زمانی سریع‌تر به پاسخ صحیح می‌رسد.

حال اگر فرض کنیم که در مثال فوق پس از وضعیت «بد» بازی به «باخت» منجر شده باشد، در این صورت یادگیری با نظارت پس از این مشاهده ارزش وضعیت جدید را به طور کامل به باخت نسبت می‌دهد، در صورتی که روش تفاضل‌های زمانی ارزش وضعیت جدید را با ۹۰٪ احتمال باخت و ۱۰٪ احتمال بُرد نسبت می‌دهد، که باز هم در این صورت این ارزش‌دهی صحیح‌تر است نسبت به یادگیری با نظارت.

توجّه به این نکته لازم است که استدلال‌های فوق تنها در صورتی صحیح هستند که محیط بازی مثال زده شده، نمونه‌ای از یک *محیط مارکوف* باشد. مثلاً اگر فرض کنیم که بازی به‌گونه‌ای طرّاحی شده باشد که وضعیت «بد» منجر به «باخت» می‌شود مگر این‌که قبل از آن وضعیت «جدید» مشاهده



شکل ۱: مثالی از یک وضعیت بازی



شکل ۲: مسئله‌ی قدم برداشتن تصادفی

شده باشد؛ این محیط، دیگر محیط مارکوف نخواهد بود، و یادگیری با نظارت پاسخ صحیح‌تری نسبت به روش تفاضل‌های زمانی ارائه خواهد داد. محیط مارکوف، محیطی است که در آن مسیر رسیدن به هدف، تنها به وضعیت عامل در هر لحظه بستگی دارد؛ نه به مسیری که عامل در گذشته برای رسیدن به آن وضعیت طی کرده است.

۵.۲ مثال قدم برداشتن تصادفی

مثال قدم *برداشتن تصادفی*[۱۱]، مثال ساده‌ی دیگری است که به کمک آن می‌توانیم کارایی پیش‌بینی در روش تفاضل‌های زمانی را بسنجیم و با روش یادگیری با نظارت مقایسه کنیم. شکل ۲، مسئله‌ی قدم برداشتن تصادفی محدود را برای ۵ گره درونی نشان می‌دهد. در این مسئله هر اپیزود از یک گره (معمولاً گره وسط؛ یا گره D در شکل) آغاز می‌شود، و سپس به احتمال ۵۰% قدم به سمت چپ و به احتمال ۵۰% قدم به سمت راست برداشته می‌شود. اگر عامل به یکی از دو وضعیت انتهایی (A یا G در شکل) برسد، اپیزود پایان می‌یابد. آن‌چه قرار است در این مسئله پیش‌بینی کنیم این است که در هر گره به چه احتمالی می‌توانیم به گره پایانی G برسیم.[۱۲]

حال نشان می‌دهیم که چگونه می‌توان از پیش‌بینی به کمک روش تفاضل‌های زمانی برای فرموله

[۱۱] Ramdom Walk

[۱۲] اگر احتمال پیش‌بینی شده برای گره i، $p(i)$ باشد، آن‌گاه احتمال رسیدن به گره A از گره i برابر خواهد بود با $۱ - p(i)$. بنابراین انتخاب گره G به عنوان گره مقصد چیزی از جامعیّت مسئله کم نمی‌کند.

کردن این مسئله و در نهایت حل کردن آن استفاده کرد. در روش تفاضل‌های زمانی نیاز به ورودی در قالب دنباله‌ی مشاهدات-نتیجه داشتیم. در این مسئله تنها برای دو گره پایانی نتیجه را می‌دانیم. احتمال رسیدن به گره G با حرکت از G مساوی یک، و با حرکت از A مساوی صفر است. بنابراین اگر دنباله‌ی گره‌های قدم زدن به G ختم شود آن‌گاه داریم $z = ۱$، و اگر این دنباله به A ختم شود آن‌گاه داریم $z = ۰$. اگر بردار مشاهده متناظر با گره i را با $\mathbf{x}_i$ نشان دهیم، آن گاه دو دنباله‌ی زیر مثال‌هایی از اپیزودهای این مسئله هستند.

$$\begin{aligned} DCDEFG \quad &\Rightarrow \quad \mathbf{x}_D, \mathbf{x}_C, \mathbf{x}_D, \mathbf{x}_E, \mathbf{x}_F, ۱ \\ DCBA \quad &\Rightarrow \quad \mathbf{x}_D, \mathbf{x}_C, \mathbf{x}_B, \mathbf{x}_A, ۰ \end{aligned}$$

جهت نمایش بردار مشاهده از بردارهای واحد پایه‌ی فضای $\mathbb{R}^۵$ می‌توانیم استفاده کنیم. مثلاً هنگامی که در گره C قرار داریم بردار مشاهده $\mathbf{x}_C = (۰, ۱, ۰, ۰, ۰)^T$ است. برای پیاده‌سازی و حلّ این مسئله نیز از روش تفاض‌های زمانی خطّی استفاده شده است، یعنی $P(\mathbf{x}_t, \mathbf{w}) = \mathbf{w}^T\mathbf{x}_t$. ترکیب این دو، محاسبات و روابط را بسیار ساده می‌کند.

برای این که مطالعه‌ی خطا و عملکرد الگوریتم‌ها از لحاظ آماری پایدار و قابل اطمینان باشد، از ۱۰۰ مجموعه‌ی ۴۰ تایی از دنباله‌های قدم برداشتن تصادفی استفاده شد و میانگین خطای پیش‌بینی[۱۳] انجام شده‌ی آن‌ها محاسبه و گزارش شده است. شکل ۳ میانگین خطای پیش‌بینی انجام شده در این مسئله را به‌ازای λ های مختلف نمایش می‌دهد. توجه کنید که به ازای $\lambda = ۱$، رابطه‌ی بروزرسانی همان رابطه‌ی Widrow-Hoff در یادگیری با نظارت است. همان‌گونه که در این شکل می‌بینیم پیش‌بینی‌های انجام شده توسط روش تفاضل‌های زمانی به‌ازای $\lambda < ۱$ از روش یادگیری با نظارت دقیق‌تر است.

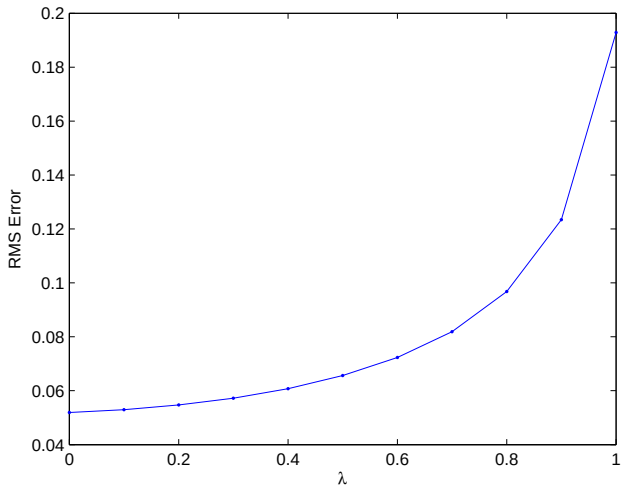
امّا مقدار نرخ یادگیری، α، نیز در خطای پیش‌بینی بی‌تأثیر نیست. در آزمایشی دیگر تأثیر مقادیر α و γ به‌طور همزمان بر خطای پیش‌بینی مورد بررسی قرار گرفته، که نتیجه‌ی آن‌را در شکل ۴ مشاهده می‌کنید. همان‌گونه که در این شکل نشان داده شده است به‌ازای انتخاب $\alpha = ۰/۱$ و $\gamma = ۰/۳$ بیشترین میزان دقّت در پیش‌بینی بدست می‌آید.

با مراجعه به بخش ۴ با برنامه‌ای که الگوریتم‌های فوق را پیاده‌سازی کرده است آشنا خواهید شد.

۳ کاربرد در یادگیری تقویتی

در این بخش به کاربرد روش تفاضل‌های زمانی در یادگیری تقویتی خواهیم پرداخت. یادگیری تقویتی زیرشاخه‌ی بزرگی از یادگیری ماشین است. در این بخش ابتدا به معرّفی یادگیری تقویتی و تعریف‌های

[۱۳] محاسبه‌ی خطای پیش‌بینی از این جهت میسّر است، که با یک تحلیل احتمالاتی ساده می‌توانیم پیش‌بینی‌های ایده‌آل را محاسبه کنیم. پیش‌بینی‌های ایده‌آل عبارتند از $\frac{۱}{۶}$، $\frac{۲}{۶}$، $\frac{۳}{۶}$، $\frac{۴}{۶}$، و $\frac{۵}{۶}$، به‌ترتیب برای گره‌های B، C، D، E، و F. رابطه‌ی خطای استفاده شده نیز، خطای RMS بین مقادیر بردار پیش‌بینی و بردار پیش‌بینی ایده‌آل می‌باشد. رابطه‌ی محاسبه‌ی خطای RMS به‌صورت $e_{rms} = \sqrt{\frac{\sum_{i=۱}^{n}(x_i - \bar{x}_i)^۲}{n}}$ است.

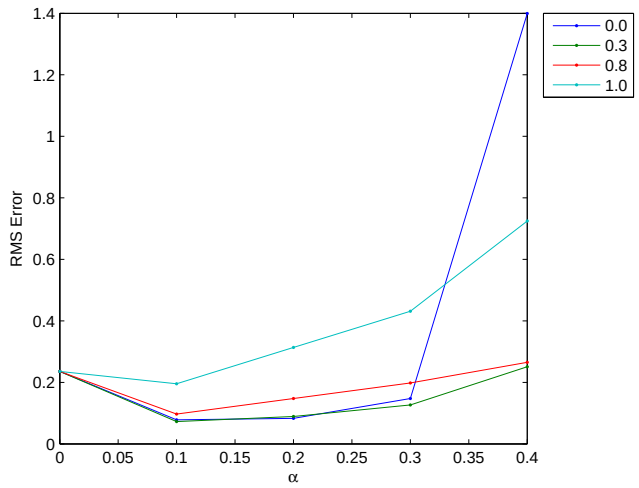


شکل ۳: میانگین خطا در مسئله‌ی پیش‌بینی قدم‌برداشتن تصادفی به‌ازای λهای مختلف

مرتبت با آن می‌پردازیم، و سپس نشان می‌دهیم که چگونه می‌توان مسئله‌ی یادگیری تقویتی را به صورت حالت خاصّی از مسئله‌ی پیش‌بینی به کمک روش تفاضل‌های زمانی در نظر گرفت و با آن این مسئله را حل کرد. پس از آن به مسئله‌ی تصمیم‌گیری در یادگیری تقویتی می‌پردازیم و نشان می‌دهیم که چگونه با تغییری بسیار جزئی می‌توان مسئله‌ی تصمیم‌گیری در یک محیط ناشناخته را حل کرد.

۱.۳ مقدّمه‌ای بر یادگیری تقویتی

یادگیری تقویتی، زیر شاخه‌ای از یادگیری ماشین است، که تمرکز آن بر یادگیری یک عامل، از طریق تعاملش با محیط می‌باشد. یادگیری تقویتی در دسته‌بندی یادگیری با نظارت قرار نمی‌گیرد؛ چرا که عامل یادگیرنده، خود از ابتدا بدون آن که ناظری عملکرد آن را به صورت صحیح یا غلط رده بندی کند، صرفاً از طریق تعامل با محیط، و پاداشی که از محیط دریافت می‌کند، عملکرد صحیح را یاد می‌گیرد. یادگیری تقویتی کاربرد بسیاری در نظریه‌ی کنترل، دارد؛ و بسیاری از محقّقانی که این شاخه را بسط و گسترش دادند، مهندسانی بودند که در زمینه‌ی کنترل تحقیق می‌کرده‌اند. در نظریه‌ی کنترل به برنامه، یا واحدی که قرار است عمل کنترل کردن را یاد بگیرد می‌گویند «کنترل‌کننده»، به سیستم مورد استفاده می‌گویند «سیستم کنترل»، به عملی که کنترل‌کننده انجام می‌دهد می‌گویند «سیگنال کنترل»، و به پاداشی که دریافت می‌کند، می‌گویند «سیگنال تقویتی». ما در این نوشتار از نامگذاری‌های به کار گرفته شده در [۱] (مرجع اوّل در یادگیری تقویتی) که عمومی‌تر است و مخاطبان بیشتری را دربر می‌گیرد استفاده می‌کنیم. زین پس به یادگیرنده می‌گوییم «عامل»، به آن‌چه که عامل با آن در تعامل است، می‌گوییم «محیط»، به عملی که عامل برای تعامل با محیط انجام می‌دهد می‌گوییم «کنش»، و به سیگنال دریافتی از محیط می‌گوییم «پاداش». در یادگیری تقویتی، هدف عامل این است که در هر وضعیت که در آن قرار می‌گیرد، طوری عمل کند که مجموع پاداش‌های مورد انتظار دریافتی از



شکل ۴: میانگین خطا در مسئله‌ی پیش‌بینی قدم‌برداشتن تصادفی به‌ازای αهای مختلف

محیط، در *دراز مدّت* بیشینه شود. وضعیتی که عامل در هر لحظه در آن قرار گرفته، از طریق بردار مشاهدات عامل در آن لحظه مشخص می‌شود.

مدل‌های مختلفی برای فرموله کردن پاداش مورد انتظار در دراز مدّت وجود دارد. یکی مدل *افق محدود*[۱۴] است، که یکی از ساده‌ترین مدل‌هایی است که می‌توان به آن فکر کرد. در این مدل، عامل باید در هر لحظه، پاداش مورد انتظار خود را در h مرحله‌ی بعدی بیشینه کند:

$$E(\sum_{t=۰}^{h-۱} r_t)$$

که در آن r_t پاداش دریافتی در t مرحله‌ی بعدی است، و E، امید ریاضی می‌باشد. چنان‌چه عملکرد عامل طوری باشد، که عبارت فوق بیشینه شود، آن گاه می‌گوییم عامل، کنش‌های بهینه‌ی h-مرحله‌ای انجام می‌دهد.

مدل دیگر، مدل *افق نامحدود تخفیف یافته*[۱۵] است. این مدل پاداش عامل در دراز مدّت را مدّنظر قرار می‌دهد، امّا پاداش‌هایی که در آینده دریافت می‌شوند، با ضریب *تخفیف* γ (که $۰ \leq \gamma < ۱$)، به طور هندسی تخفیف داده می‌شوند:

$$E(\sum_{t=۰}^{\infty} \gamma^t r_t)$$

وجود ضریب γ موجب می‌شود که پاداش آنی، وزن بیشتری نسبت به پاداش‌های مورد انتظار در آینده داشته باشد، و در عین حال پاداش‌های آینده نیز نادیده گرفته نشوند. هرچه γ به ۱ نزدیک‌تر

[۱۴] Finite Horizon

[۱۵] Infinite Horizon Discounted

باشد، الگوریتم یادگیری آینده‌نگرتر می‌شود [۵]. با انتخاب $\gamma = ۰$، الگوریتم یادگیری، تبدیل به یک الگوریتم حریص می‌شود که تنها پاداش آنی را بیشینه می‌کند. کاربرد دیگر ضریب γ، کران‌دار کردن سری فوق است. از محدود بودن مقدار این سری در اثبات قضایای همگرایی یادگیری تقویتی استفاده می‌شود [۵، ۶]. به همین دلیل است که انتخاب $\gamma = ۱$، غیرمجاز است، مگر در سناریوهای اپیزودیک، که در آن هر اپیزود پس از طی تعداد محدودی مرحله به پایان می‌رسد.

دو مدل ارائه شده در بالا تنها مدل‌های فرموله کردن پاداش مورد انتظار در دراز مدّت نیستند، امّا مهم‌ترین و رایج‌ترین‌شان هستند، و ما در این نوشتار تنها با این دو مدل سروکار داریم.

۲.۳ استفاده از روش تفاضل‌های زمانی در یادگیری تقویتی

همان‌گونه که پیش از این گفته شد، هدف روش تفاضل‌های زمانی، پیش‌بینی چند مرحله‌ای با استفاده از ورودی دنباله‌ی مشاهدات-نتیجه است. در این قسمت نشان می‌دهیم که چگونه مسئله‌ی یادگیری تقویتی را می‌توان به صورت یک حالت خاص روش تفاضل‌های زمانی در نظر گرفت، و به کمک این روش، این مسئله را حل کرد.

یک عامل را فرض کنید که در هر لحظه‌ی t، به عنوان ورودی زوج مرتب $(\mathbf{x}_t, r_t)$ را دریافت می‌کند، که در آن $\mathbf{x}_t$، بردار مشاهدات عامل در لحظه‌ی t، و r_t یک عدد حقیقی است که معرّف پاداشی است که عامل در لحظه‌ی t از محیط دریافت می‌کند. بردار مشاهدات، $\mathbf{x}_t$، برای عامل می‌تواند صرفاً از پارامترهایی تشکیل شود، که حالت محیط را نشان می‌دهند، یا علاوه بر آن شامل کنشی که عامل در لحظه‌ی t انجام داد نیز باشد:

$$\mathbf{x}_t = \mathbf{s}_t \quad \text{یا} \quad \mathbf{x}_t = \langle \mathbf{s}_t, a_t \rangle$$

این انتخاب به صورت مسئله بستگی دارد. اگر هدف مسئله، یافتن کنش بهینه در هر وضعیت باشد؛ بهتر است، کنش را نیز قسمتی از بردار مشاهدات در نظر بگیریم، امّا اگر هدف مسئله، صرفاً یافتن وضعیت بهینه برای رسیدن به هدف باشد، و این که بخواهیم وضعیت‌های مختلف محیط را از این نظر وزن‌دهی کنیم، بنابراین بهتر است بردار مشاهدات، تنها شامل بردار وضعیت باشد.

با استفاده از روش تفاضل‌های زمانی، هدف ما این است که با دریافت دنباله‌های $(\mathbf{x}_t, r_t)$، برای $t = ۰, ۱, \ldots$، در هر لحظه‌ی t، پیش‌بینی P_t از کمیت زیر را انجام دهیم:

$$R_t = r_{t+۱} + \gamma r_{t+۲} + \gamma^۲ r_{t+۳} + \ldots = \sum_{i=۱}^{\infty} \gamma^{i-۱} r_{t+i}$$

تخمین انجام شده، یک پیش‌بینی است، چرا که مقادیر r، در آینده را دربر می‌گیرد. پیش‌بینی P_t، تابعی است از بردار مشاهدات $\mathbf{x}_t$، که آن را به صورت $P_t = P(\mathbf{x}_t)$ نشان می‌دهیم، که در آن P تابع پیش‌بینی نام دارد. به نقش γ در رابطه‌ی R_t، توجه کنید. اگر $\gamma = ۰$ قرار دهیم هدف از الگوریتم یادگیری، تنها پیش‌بینی $R_t = r_{t+۱}$ می‌شود.

البته همواره یافتن تابع پیش‌بینی که این مسئله را بخوبی حل کند ممکن نیست. اگر هیچ گونه نظمی یا رابطه‌ای بین بردار $\mathbf{x}_t$ و مقادیر r_t در آینده وجود نداشته باشد، آن‌گاه بهترین پیش‌بینی‌ها

بهتر از حدس زدن تصادفی نخواهد بود. امّا اگر نظمی در این مقادیر وجود داشته باشد، مثلاً اگر مقادیری که قرار است پیش‌بینی شوند حاصل تحوّل یک سیستم پویا باشند، روش تفاضل‌های زمانی این مقادیر را به خوبی پیش‌بینی می‌کنند [۲]. بخصوص اگر مقادیر $\mathbf{x}_t$ معرّف وضعیت‌های یک زنجیره‌ی مارکوف باشند، بسته به این که بردار مشاهدات آیا به طور دقیق نشان‌دهنده‌ی وضعیت‌های زنجیره‌ی مارکوف هستند، یا که برخی از وضعیت‌ها قابل مشاهده نبوده‌اند، یک تابع پیش‌بینی ممکن است وجود داشته باشد که به طور دقیق مقادیر مورد انتظار R_t را برای هر t بدست می‌دهد [۷].

هدف در روش تفاضل‌های زمانی، یادگیری مقادیر بردار وزن‌ها می‌باشد. طول بردار وزن‌ها متناسب است با طول بردار مشاهدات، بخصوص در صورت استفاده از روش تفاضل‌های زمانی خطّی، طول بردار وزن‌ها باید با طول بردار مشاهدات برابر باشد. در یادگیری تقویتی در صورتی که بردار مشاهدات صرفاً شامل وضعیت محیط باشد، به بردار وزن‌ها می‌گویند *تابع ارزش* و آن را با V نمایش می‌دهند. در صورتی که بردار مشاهدات علاوه بر وضعیت محیط، شامل کنش نیز باشد، آن را با Q نمایش می‌دهند.

اکنون می‌خواهیم با یک مثال ساده، طرز استفاده از روش تفاضل‌های زمانی را در یادگیری تقویتی شرح دهیم. فرض کنید در هر لحظه محیط دارای وضعیت s، باشد، که s یک عضو از فضای حالات $\mathcal{S}$ می‌باشد. اگر فضای حالات را گسسته و محدود فرض کنیم، آن‌گاه می‌توانیم بازای هر حالت در فضای حالات، یک درایه در بردار وزن‌ها در نظر بگیریم. در این صورت بدست آوردن ارزش هر حالت به سادگی جستجو در یک جدول خواهد بود. بدین ترتیب طول بردار وزن‌ها، به بزرگی اندازه‌ی فضای حالت خواهد شد:

$$|V| = n(\mathcal{S})$$

بردار مشاهده را نیز طوری در نظر می‌گیریم که فقط مشخّص کند، کدام حالت در فضای حالات مشاهده شده است. مثلاً می‌توان از برداری استفاده کرد که تمام درایه‌های آن صفر است، بجز یک درایه که متناظر است با حالت مشاهده شده. بنابراین هر بردار مشاهده را، یک پایه برای فضای $\mathbb{R}^{n(\mathcal{S})}$ در نظر می‌گیریم.

برای سادگی ابتدا از روش TD(0) خطّی استفاده می‌کنیم، و γ را مساوی صفر فرض می‌کنیم؛ یعنی قصد داریم تنها یک مرحله جلوتر را پیش‌بینی کنیم، و $P_t = r_{t+1}$. با توجه به بحثی که در بخش ۳.۲ صورت گرفت و رابطه‌ی (۶) برای بروز رسانی وزن‌ها در روش TD(0)، یعنی:

$$\Delta V_t = \alpha(P_{t+1} - P_t)\nabla_V P_t \tag{۷}$$

می‌توان روابط زیر را نتیجه گرفت. روش TD(0) خطّی است، پس گرادیان فوق بصورت زیر کاهش می‌یابد:

$$P_t = V^T\mathbf{x}_t \Rightarrow \nabla_V P_t = \mathbf{x}_t$$

امّا بردار مشاهده، $\mathbf{x}_t$، برداری است که تمام درایه‌های آن صفر است، به‌جز یک درایه که متناظر است با وضعیت محیط، در لحظه‌ی t. مثلاً اگر وضعیت محیط در لحظه‌ی t، همان وضعیت شماره‌ی sام باشد در فضای حالت $\mathcal{S}$، تنها درایه‌ی sام بردار $\mathbf{x}_t$، مساوی یک است. بنابراین رابطه‌ی بروز رسانی (۷) بصورت زیر کاهش می‌یابد:

$$V_{t+1}(s) \leftarrow V_t(s) + \alpha(P_{t+1} - P_t) \tag{۸}$$

که در آن منظور از $V(s)$، درایه‌ی sام بردار V است؛ و منظور از V_t، نسخه‌ای از بردار V است که در زمان t، در دسترس بود. ما قصد داریم الگوریتم یادگیری ما جوری عمل کند که در انتها داشته باشیم $P_{t+1} = r_{t+1}$، و همچنین داریم:

$$P_t = P_t(\mathbf{x}_t) = V^T \mathbf{x}_t = V(s)$$

نتیجه‌گیری جالب این است که تابع پیش‌بینی همان بردار ارزش‌ها است، و با یادگیری آن به طور خودبه‌خود تابع پیش‌بینی را نیز یاد گرفته‌ایم، و در دراز مدّت مقادیر این بردار برابر خواهند بود با مقادیر مورد انتظار پاداش. با جمع‌بندی از مطالب فوق نهایتاً رابطه‌ی بروزرسانی به صورت زیر در می‌آید:

$$V_{t+1}(s) \leftarrow V_t(s) + \alpha(r_{t+1} - V_t(s)) \tag{۹}$$

که در آن s نماینده‌ی وضعیت محیط در لحظه‌ی t است. ظاهر رابطه‌ی فوق بیانگر این است، که چنان‌چه بخواهیم مقادیر بردار ارزش‌ها را به‌ازای وضعیت s بروزرسانی کنیم، باید تا لحظه‌ی $t+1$ صبر کنیم تا پاداش r_{t+1} را دریافت کنیم. به عبارت دیگر، این الگوریتم در هر لحظه‌ی t، مقادیر بردار ارزش‌ها در گام زمانی قبلی را بروز می‌کند.

اگر بخواهیم حالت کلّی‌تری را در نظر بگیریم، که در آن هدف از یادگیری این است که مقادیر بردار V، مقدار R_t را پیش‌بینی کنند آن‌گاه رابطه‌ی بروز رسانی فوق حالت خاصّی از رابطه‌ی زیر خواهد بود:

$$V_{t+1}(s) \leftarrow V_t(s) + \alpha(R_t - V_t(s)) \tag{۱۰}$$

که در آن:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots$$

اگر بخواهیم از رابطه‌ی فوق به‌ازای $\gamma \neq 0$ استفاده کنیم، بروزرسانی به‌ازای زمان t، باید در بی‌نهایت انجام شود، چرا که مقدار R_t، در بی‌نهایت مشخّص می‌شود. بنابراین با این فرض رابطه‌ی فوق، یک قانون یادگیری مناسب را بدست نمی‌دهد، امّا مشاهده‌ی زیر می‌تواند سودمند باشد:

$$\begin{aligned} R_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \\ &= r_{t+1} + \gamma \left(r_{t+2} + \gamma r_{t+3} + \ldots\right) \\ &= r_{t+1} + \gamma R_{t+1} \end{aligned}$$

اگر وضعیت بعد از s را در محیط با s' نشان دهیم، آن‌گاه می‌دانیم که $V_t(s')$ تخمینی برای R_{t+1} است در لحظه‌ی t. بنابراین می‌توانیم R_t را با $r_{t+1} + \gamma V_t(s')$ تخمین بزنیم، و در رابطه‌ی بروز رسانی بردار ارزش‌ها از آن استفاده نماییم:

$$V_{t+1}(s) \leftarrow V_t(s) + \alpha(r_{t+1} + \gamma V_t(s') - V_t(s)) \tag{۱۱}$$

این که وضعیت s'، چه می‌تواند باشد، بستگی به کنش و وضعیت عامل در لحظه‌ی t، دارد. نگاشتی که مشخّص می‌کند، عامل در هر وضعیت، ممکن است چه کنشی را انجام دهد را سیاست[۱۶] عامل می‌گویند، و آن را با π نشان می‌دهند. مسلّماً کیفیت یادگیری مقادیر V، بستگی به سیاست عامل دارد. مثلاً اگر عاملی در تمامی وضعیت‌ها تنها زیرمجموعه‌ی کوچکی از کنش‌های ممکن را انجام دهد، آن‌گاه بردار V، برای وضعیت‌هایی که هیچ‌گاه تجربه نشدند، مقادیر دقیقی نخواهد داشت. به خاطر اهمیّت زیادی که سیاست عامل در صحّت مقادیر V دارد، گاهی آن را با V^π نمایش می‌دهند، تا آن را با V ایده‌آل که بدنبال آن هستیم متمایز کنند.

الگوریتم ۳ بکارگیری روش TD(0) را در یادگیری تقویتی نشان می‌دهد.

الگوریتم ۳ الگوریتم استفاده از TD(0) در یادگیری تقویتی

```
Initialize V arbitrarily                          ▷ بردار ارزش‌ها را به‌طور دلخواه مقداردهی اوّلیه کن.
for all episode do                                ▷ برای هر اپیزود
    Initialize s
    for all step of episode do
        a ← action given by π for s
        Take action a;                            ▷ کنش بدست آمده از طریق π را انجام بده
        observe r, and next state s'              ▷ پاداش و وضعیت بعدی محیط را مشاهده کن
        V(s) ← V(s) + α [r + γV(s') − V(s)]
        s ← s'
    end for
end for
```

۳.۳ مسئله‌ی تصمیم‌گیری

اصلی‌ترین مسئله‌ای که در یادگیری تقویتی به دنبال حل کردن آن هستیم، مسئله‌ی تصمیم‌گیری است. مسئله‌ی تصمیم‌گیری این سوال را مطرح می‌کند که عامل در هر وضعیت چه کنشی را باید انجام دهد. مسئله‌ای که در قسمت قبل حل کردیم، صرفاً وضعیت‌ها را از نظر پاداش مورد انتظار در آینده وزن‌دهی می‌کرد؛ امّا به این سوال پاسخ نمی‌داد که کنش بهینه در هر وضعیت چیست؟

برای پاسخ به این سوال یک تغییر جزئی در راه حل ما را به هدف بسیار نزدیک می‌کند. تغییری که باید اعمال شود این است که بردار مشاهدات در لحظه‌ی t، علاوه بر وضعیت محیط در لحظه‌ی t، شامل کنش انجام شده در این لحظه نیز باشد؛ یعنی

$$\mathbf{x}_t = \langle \mathbf{s}_t, a_t \rangle$$

[۱۶] Policy

با اعمال این تغییر، آن‌چه که عملاً وزن‌دهی می‌شود، مطلوبیت وضعیت‌ها نیست، بلکه مطلوبیت زوج‌های وضعیت-کنش است. مرسوم است، که چنان‌چه از زوج وضعیت-کنش استفاده شود؛ به‌جای V، بردار وزن‌ها را Q بنامیم. کیفیت یادگیری مقادیر Q، به عملکرد عامل در محیط بستگی دارد. عامل باید سعی کند به گونه‌ای در محیط رفتار کند که زوج‌های وضعیت-کنش به تعداد کافی تجربه شوند. برای این منظور عامل باید سعی کند تا آن‌جا که ممکن است محیط پیرامون خود را کشف کند. از طرفی دیگر عامل باید پاداش‌های دریافتی از محیط را در درازمدّت بیشینه کند. اکنون سؤالی که مطرح می‌شود این است که، عامل چگونه می‌تواند این دو هدف به ظاهر متناقض را با هم برآورده سازد؟

یک راه برآورده کردن این هدف در طولانی‌مدّت استفاده از *سیاست اپسیلون-حریصانه*[۱۷] است؛ سیاستی که در آن عامل به احتمال ϵ کنش تصادفی را انجام می‌دهد، و در باقی مواقع کنشی را انجام می‌دهد که در بردار Q بیشترین ارزش را بخود اختصاص داده. هرچه اپسیلون بزرگ‌تر باشد، عامل زمان بیشتری را صرف تجربه کردن محیط ناشناخته‌ی اطراف خود می‌کند؛ و هر چه اپسیلون کوچک‌تر باشد، عامل بیشتر پیرو سیاست بهینه‌ای است که مقادیر Q آن را نشان می‌دهند. کنش تصادفی که به احتمال ϵ انجام می‌شود، همان است که به عامل در شناختن بهتر محیط اطرافش کمک می‌کند.

یک استراتژی مناسب این است که در ابتدای یادگیری، مقدار ϵ را بزرگ اختیار کنیم. هرچه عامل بیشتر و بیشتر به مقادیر ایده‌آل Q نزدیک شد مقدار ϵ را به صفر میل دهیم. مشکل این استراتژی این است که تشخیص این که آیا عمل یادگیری انجام شده یا خیر، خود بسیار دشوار است. در تمامی مثال‌هایی که در این نوشتار ارائه شده‌اند؛ مقدار ϵ ثابت و مساوی ۰/۱ فرض شده است.

الگوریتم یادگیری تصمیم‌گیری، که معادل الگوریتم ۳، امّا با درنظر گرفتن زوج وضعیت-کنش به جای وضعیت به‌تنهایی است، در الگوریتم ۴ ارائه شده است. این الگوریتم به نام SARSA[۱۸] معروف است.

الگوریتم SARSA یک الگوریتم یادگیری *مبتنی بر سیاست*[۱۹] است. بدین معنی که در نهایت پاسخی ارائه می‌دهد که برای سیاستی که عامل در پیش گرفته بهینه است. گاهی این پاسخ با جواب بهینه‌ی مسئله متفاوت است. الگوریتم یادگیری که پاسخ بهینه‌ی مسئله را صرف‌نظر از سیاستی که عامل در پیش‌گرفته بدست دهد را یک الگوریتم یادگیری *مستقل از سیاست*[۲۰] می‌گویند. الگوریتم یادگیری Q-Learning (الگوریتم ۵) نمونه‌ای از یک الگوریتم یادگیری مستقل از سیاست است؛ که علّت آن را در رابطه‌ی بروزرسانی مقادیر Q در این الگوریتم می‌توان جویا شد. برای درک بهتر از تفاوت این دو نوع الگوریتم به مثال *راه‌رفتن کنار درّه* در بخش مثال‌ها، بخش ۴.۴.۳، مراجعه کنید.

---

[۱۷] ϵ-greedy policy

[۱۸] کلمه‌ی SARSA مخفّف State-Action-Reward-State-Action است؛ و بر اهمّیت این توالی در مسئله‌ی یادگیری تصمیم‌گیری تأکید دارد. این توالی در رابطه‌ی به‌روزرسانی Q در خط ۸ الگوریتم ۴ مشهود است.

[۱۹] on-policy

[۲۰] off-policy

الگوریتم ۴ الگوریتم یادگیری تصمیم‌گیری SARSA

```
Initialize Q(s, a) arbitrarily
for all episodes do
    Initialize s
    Choose a from s using policy derived from Q (e.g., ϵ-greedy)
    for all step of episode do
        Take action a, observe r, s′
        Choose a′ from s′ using policy derived from Q (e.g., ϵ-greedy)
        Q(s, a) ← Q(s, a) + α [r + γQ(s′, a′) − Q(s, a)]
        s ← s′; a ← a′
    end for
end for
```

الگوریتم ۵ الگوریتم یادگیری تصمیم‌گیری Q-Learning

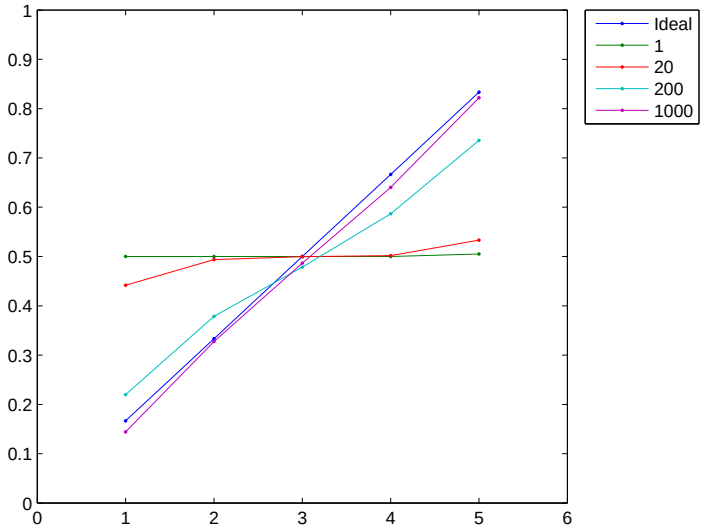
```
Initialize Q(s, a) arbitrarily
for all episode do
    Initialize s
    for all step of episode do
        Choose a from s using policy derived from Q (e.g., ϵ-greedy)
        Take action a, observe r, s′
        Q(s, a) ← Q(s, a) + α [r + γ max_a′ Q(s′, a′) − Q(s, a)]
        s ← s′;
    end for
end for
```

۴.۳ مثال‌ها

در این قسمت مثال‌هایی از به‌کارگیری الگویتم‌های یادگیری تقویتی در مسائل مختلف نشان داده می‌شود. مثال‌هایی که مشاهده خواهید کرد در [۱] پیش‌نهاد شده‌اند. در این بخش تنها به بررسی عملکرد الگوریتم‌های یادگیری‌ای که قبلاً معرّفی شده‌اند خواهیم پرداخت. برای اطلاعات بیشتر درباره‌ی پیاده‌سازی و راهنمای استفاده از برنامه‌ی نوشته شده به بخش ۴ و برنامه‌های ارائه شده به همراه این نوشتار مراجعه کنید.

۱.۴.۳ قدم برداشتن تصادفی

در بخش ۵.۲ با مسئله‌ی قدم برداشتن تصادفی و حلّ آن به کمک روش تفاضل‌های زمانی با کاربرد در مسئله‌ی پیش‌بینی آشنا شدید. در این قسمت قصد داریم با حلّ همان مسئله امّا با رهیافت یادگیری



شکل ۵: حلّ مسئله‌ی قدم برداشتن تصادفی با ۵ گره درونی با استفاده از یادگیری تقویتی

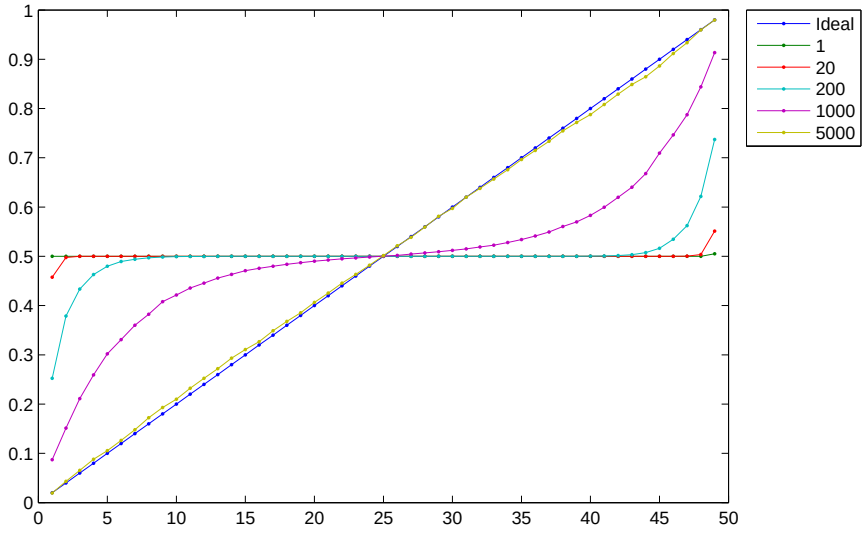
تقویتی آشنا شویم. ابتدا این مسئله را با ۵ گره درونی حل می‌کنیم؛ درست همانند مسئله‌ای که در بخش ۵.۲ به حلّ آن پرداختیم. برای حلّ این مسئله از الگوریتم ۳، کاربرد TD(0) در یادگیری تقویتی استفاده شده است. منظور از وضعیت در این الگوریتم، همان گره‌های قدم برداشتن تصادفی است؛ و منظور از سیاست در این الگوریتم همان قدم برداشتن تصادفی است، یعنی اگر در لحظه‌ی t در یک گره درونی قرار گرفته باشیم در لحظه‌ی $t+۱$ به احتمال ۵۰٪ به سمت چپ حرکت می‌کنیم، و به احتمال ۵۰٪ به سمت راست. با این توضیحات این الگوریتم برای ۱۰۰۰ اپیزود با نرخ یادگیری، $\alpha = ۰/۰۱$، و ضریب تخفیف، $\gamma = ۱$، اجرا شده است، که نتیجه‌ی اجرای آن را برای اپیزودهای، ۱، ۲۰، ۲۰۰، و ۱۰۰۰ و مقایسه‌ی آن با پیش‌بینی ایده‌آل در شکل ۵ مشاهده می‌نمایید.

همچنین در شکل ۶ حاصل اجرای همین الگوریتم برای مسئله‌ی قدم برداشتن تصادفی برای ۴۹ گره درونی، را برای اپیزودهای ۱، ۲۰، ۲۰۰، ۱۰۰۰، و ۵۰۰۰ را مشاهده می‌نمایید. در این شکل به خوبی سیر همگرایی پیش‌بینی‌های انجام شده به پیش‌بینی ایده‌آل، قابل مشاهده است.

۲.۴.۳ جهان مشبّک

همان‌طور که در شکل ۷ مشاهده می‌نمایید، *جهان مشبّک*[۲۱] محیطی است مشبّک که در آن عامل از خانه‌ی آغازین شروع به حرکت می‌کند، و هدف آن رسیدن به خانه‌ی هدف است. در تمامی مثال‌های این بخش خانه‌ی آغازین را با S و خانه‌ی هدف را با G نمایش می‌دهیم. ابتدا فرض می‌کنیم که عامل، تنها قادر است در چهار جهت اصلی (بالا، پایین، چپ، و راست) در این محیط حرکت کند. این مسئله با استفاده از الگوریتم یادگیری SARSA و سیاست اپسیلون-حریصانه با $\epsilon = ۰/۱$ حل

[۲۱] Grid World



شکل ۶: حلّ مسئله‌ی قدم برداشتن تصادفی با ۴۹ گره درونی با استفاده از یادگیری تقویتی

شده است. در تمامی مثال‌های جهان مشبّک، به ازای هر حرکتی که عامل انجام می‌دهد پاداش ۱− می‌گیرد و در صورت رسیدن به هدف پاداش ۰ می‌گیرد. هدف از این نحوه‌ی طرّاحی پاداش‌ها این است که ما بدنبال کوتاه‌ترین مسیر می‌گردیم.

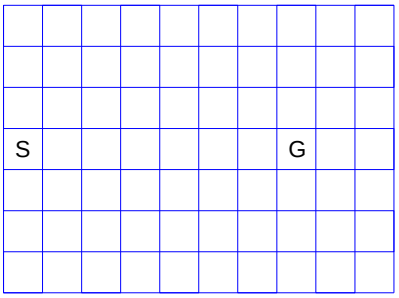
برای حلّ این مسئله از الگوریتم یادگیری SARSA با نرخ یادگیری $\alpha = ۰/۱$ و ضریب تخفیف $\gamma = ۰/۹$ در یک محیط مشبّک ۷ × ۱۰ استفاده شد، که نتیجه‌ی اجرای آن را در شکل ۸ مشاهده می‌نمایید. همچنین حاصل یادگیری این الگوریتم در یک محیط ۲۰ × ۲۰، را که در آن عامل علاوه بر چهار جهت اصلی می‌تواند حرکت قطری هم انجام دهد (حرکت شاه) در شکل ۹ نشان داده شده است.

۳.۴.۳ جهان مشبّک طوفانی

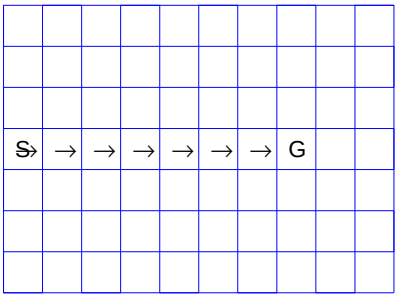
جهان مشبّک طوفانی، نمونه‌ای از مسئله‌ی جهان مشبّک است که در هر ستون آن باد با شدّتی متفاوت از ستون دیگر و در جهت عمودی می‌وزد. در صورتی که عامل به یکی از خانه‌ها حرکت کند، به اندازه‌ی شدّت باد در جهت عمودی جابجا می‌شود. این یک مثال بسیار خوب از کاربرد یادگیری تقویتی در مسائلی است که در آن محیط ناشناخته است[۲۲] و نتیجه‌ی هر کنش عامل به خوبی از قبل قابل پیش‌بینی نیست[۲۳]. در شکل ۱۰ حاصل اعمال الگوریتم یادگیری SARSA در این محیط را مشاهده می‌نمایید، هم‌چنین شکل ۱۱ حلّ همان مسئله را در صورتی که عامل بتواند حرکت شاه انجام دهد را نشان می‌دهد.

[۲۲] عامل از شدّت باد آگاهی ندارد

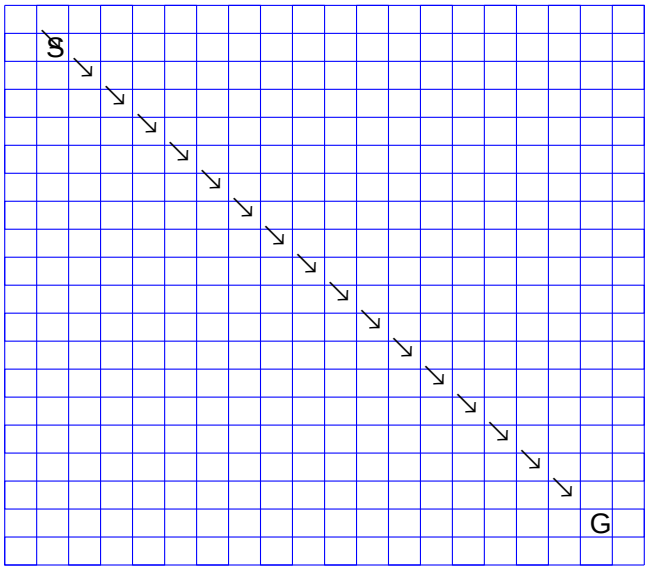
[۲۳] مثلاً حرکت عامل به سمت راست لزوماً او را به خانه‌ی سمت راست نمی‌برد



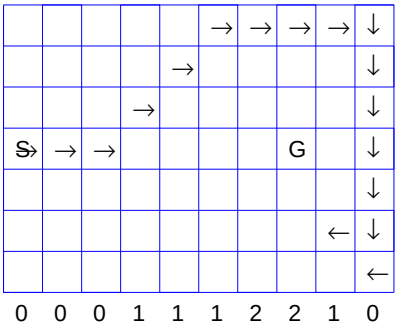
شکل ۷: مسئله‌ی جهان مشبّک



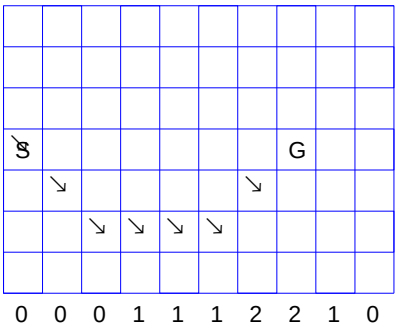
شکل ۸: حلّ مسئله‌ی جهان مشبّک ۷ × ۱۰



شکل ۹: حلّ مسئله‌ی جهان مشبّک ۲۰ × ۲۰، با حرکت شاه



شکل ۱۰: حلّ مسئله‌ی جهان مشبّک طوفانی



شکل ۱۱: حلّ مسئله‌ی جهان مشبّک طوفانی، با حرکت شاه

۴.۴.۳ راه‌رفتن کنار درّه

مسئله‌ی *راه رفتن کنار درّه*[۲۴] از مسائل خوبی است که تفاوت الگوریتم‌های یادگیری SARSA و Q-Learning را بخوبی نشان می‌دهد. در محیط این مسئله ناحیه‌ای در نظر گرفته می‌شود، به نام درّه، که چنان‌چه عامل به یکی از خانه‌های درّه برود، پاداش ۱۰۰− می‌گیرد (در واقع این پاداش، یک تنبیه بزرگ است). در شکل‌های نشان داده شده، خانه‌های درّه را با حرف C نشان می‌دهیم. حاصل استفاده از الگوریتم مبتنی بر سیاست SARSA، برای حلّ این مسئله در شکل ۱۲ نشان داده شده است؛ و حاصل استفاده از الگوریتم مستقل از سیاست Q-Learning نیز در شکل ۱۳ نشان داده شده است. در نگاه اوّل به نظر می‌آید که پاسخ صحیح‌تر، پاسخی است که الگوریتم Q-Learning ارائه می‌دهد، چرا که کوتاهترین مسیر را نشان می‌دهد. در واقع الگوریتم Q-Learning یک الگوریتم مستقل از سیاست است و پاسخ بهینه برای مسئله را بدست می‌دهد، نه پاسخ بهینه برای عامل. پاسخ بهینه برای عامل همان پاسخی است که الگوریتم SARSA ارائه داده است. علّت این ادّعا این است که سیاست عامل ϵ-حریصانه است، با مقدار $\epsilon = ۰/۱$؛ بدین معنی که عامل در ۱۰ درصد مواقع از مسیر بهینه‌ی پیش‌نهادی (نشان داده شده با پیکان در شکل) پیروی نمی‌کند و یک عمل اتّفاقی انجام می‌دهد تا بتواند محیط پیرامون خود را کشف کند. حال اگر عامل کنار درّه حرکت کند آن‌گاه ممکن است در اثر انجام عمل اتّفاقی خود را به درون درّه بیاندازد. برای همین الگوریتم SARSA مسیر حرکت از یک سطر بالاتر را پیش‌نهاد می‌دهد تا در صورت انجام عمل اتّفاقی عامل به درون درّه نیفتد.

[۲۴] Cliff-Walking

→	→	→	→	→	→	→	→	→	↓
↑									↓
$	C	C	C	C	C	C	C	C	G

شکل ۱۲: حلّ مسئله‌ی راه رفتن کنار درّه با استفاده از الگوریتم SARSA

→	→	→	→	→	→	→	→	→	↓
$	C	C	C	C	C	C	C	C	G

شکل ۱۳: حلّ مسئله‌ی راه رفتن کنار درّه با استفاده از الگوریتم Q-Learning

۴ نرم‌افزار طرّاحی شده

در این بخش به معرّفی نرم‌افزار طرّاحی شده برای به نمایش گذاشتن مثال‌هایی از روش تفاضل‌های زمانی می‌پردازیم. این نرم‌افزار به زبان Matlab پیاده سازی شده است. برای اجرای رابط گرافیکی اصلی این نرم‌افزار که به کمک آن می‌توانید تمامی مثال‌ها را با پارامترهای متنوّع اجرا کنید، به پوشه‌ی مثال‌ها بروید و فایل DemoGUI.m را اجرا کنید. پنجره‌ی اصلی برنامه در شکل ۱۴ دیده می‌شود، که به کمک آن می‌توانید مثال‌های مختلف را با پارامترهای دلخواه اجرا کنید. در بالای پنجره یک لیست پایین افتادنی از مثال‌های آماده وجود دارد، تحت عنوان Predefined Demos. این‌ها هریک نمونه‌ای از مثال‌های مختلف هستند که پارامترهایش از قبل تعیین شده است. می‌توانید هر یک از این مثال‌ها را انتخاب کنید، و پارامترهای مرتبط با آن را تغییر دهید.

توجّه: توصیه می‌شود قبل از هر چیز، مثال‌های تهیه شده در قسمت Predefined Demos را اجرا کنید، تا با نحوه‌ی کار برنامه به طور کلّی آشنا شوید. البتّه از طریق گزینه‌هایی که زیر آن قرار گرفته‌اند، همواره می‌توانید مثال‌های از قبل تهیه شده را باز تولید نمایید.

در رابط گرافیکی اصلی برنامه، با استفاده از لیست پایین‌آمدنی با عنوان Select Problem Type نوع مسئله را می‌توانید انتخاب کنید. با انتخاب هر مسئله پنل مربوط به پارامترهای آن مسئله فعّال می‌شود. مسئله‌های قابل انتخاب از این طریق عبارتند از:

۱. مسئله‌ی پیش‌بینی قدم برداشتن تصادفی

۲. حلّ مسئله‌ی قدم برداشتن تصادفی به روش یادگیری تقویتی

۳. مسائل جهان مشبّک

با غیر فعّال کردن گزینه‌ی Show Plot Title، عنوان نمودارهایی که برنامه تولید می‌کند حذف خواهند شد. از این گزینه برای تولید نمودارها و شکل‌های این نوشتار استفاده شده است.

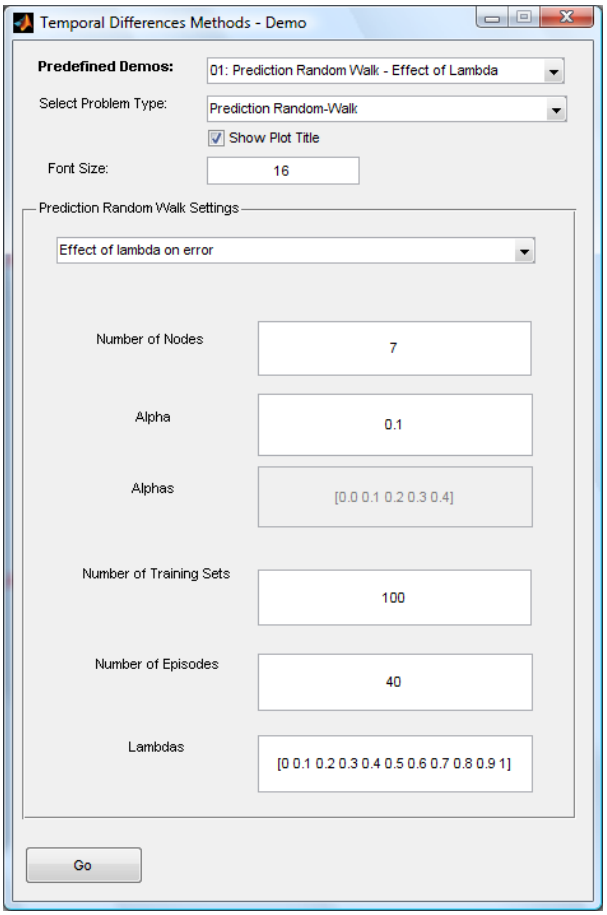
گزینه‌ی Font Size اندازه‌ی فونت استفاده شده در رسم شکل‌های جهان مشبّک را مشخّص می‌کند. در صورتی که ابعاد مسئله را بزرگ یا کوچک کنید، می‌توانید با کوچک یا بزرگ کردن اندازه‌ی فونت، شکل واضح‌تر و گویاتری از وضعیت جهان مشبّک بدست آورید. این گزینه نیز جهت تسهیل در تولید شکل‌های این نوشتار به برنامه اضافه شده است.

در صورتی که نوع مسئله را «مسئله‌ی پیش‌بینی قدم برداشتن تصادفی» یا Prediction Random-Walk انتخاب نمایید، پنل پارامترهای این نوع مسئله فعّال می‌شود. مسئله‌ی پیش‌بینی قدم برداشتن تصادفی خود به دو دسته تقسیم می‌شود:

۱. بررسی تأثیر مقدار λ بر دقّت پیش‌بینی

۲. بررسی تأثیر مقادیر λ و α به طور همزمان بر دقّت پیش‌بینی

گزینه‌ی Number of Nodes، معرّف تعداد کلّ گره‌های این مسئله است. این تعداد شامل گره پایانی سمت راست و گره پایانی سمت چپ نیز می‌شود. بنابراین برای حلّ مثال مطرح شده در این نوشتار، که در آن نیاز به مدلی با ۵ گره *درونی* می‌باشد، باید تعداد کلّ گره‌ها ۷ وارد شود.



شکل ۱۴: رابط گرافیکی اصلی برنامه

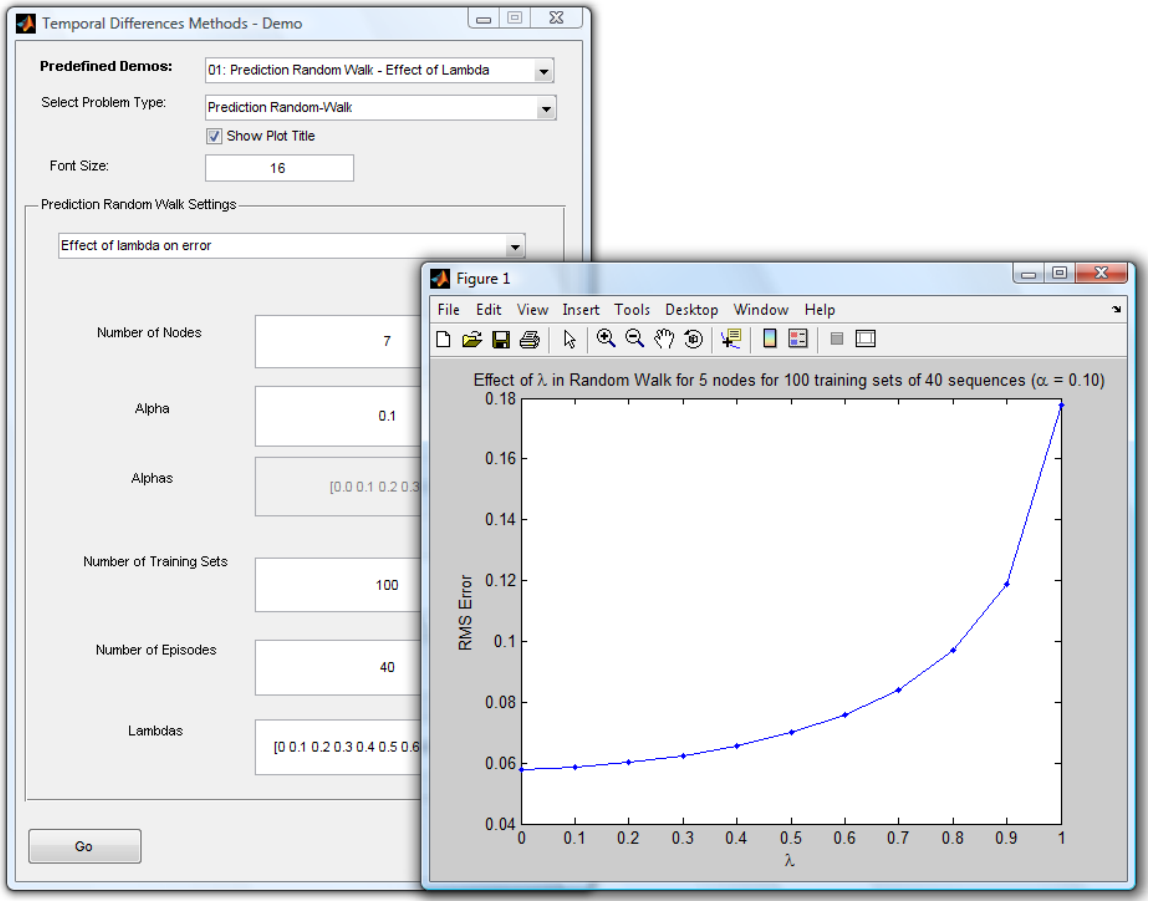
گزینه‌ی Alpha مقدار α را برای مسئله‌ی «بررسی تأثیر مقدار λ بر دقّت پیش‌بینی» تعیین می‌کند. در صورتی که نوع مسئله را «بررسی تأثیر مقادیر λ و α به طور همزمان بر دقّت پیش‌بینی» انتخاب نمایید، این گزینه غیرفعّال می‌شود.

گزینه‌ی Alphas مقادیر مختلف α را برای مسئله‌ی «بررسی تأثیر مقادیر λ و α به طور همزمان بر دقّت پیش‌بینی» تعیین می‌کند. در صورتی که نوع مسئله را «بررسی تأثیر مقدار λ بر دقّت پیش‌بینی» انتخاب نمایید، این گزینه غیرفعّال می‌شود.

گزینه‌ی Number of Training Sets تعداد مجموعه‌های آموزشی را تعیین می‌کند. هر مجموعه‌ی آموزشی شامل تعدادی اپیزود مسئله‌ی قدم برداشتن تصادفی است. به طوری که تمامی اپیزودهای داخل مجموعه‌ی آموزشی یکی پس از دیگری به یادگیرنده ارائه می‌شوند و در نهایت بردار وزن‌های آن مجموعه‌ی آموزشی مورد بررسی قرار می‌گیرد. در انتها، میانگین خطای همه‌ی مجموعه‌های آموزشی گزارش می‌شود.

گزینه‌ی Number of Episodes تعداد اپیزودهای داخل هر مجموعه‌ی آموزشی را تعیین می‌کند.

گزینه‌ی Lambdas مقادیر مختلف λ در هر دو مسئله را تعیین می‌کند.
شکل ۱۵ اجرای نمونه‌ای از مسئله‌ی پیش‌بینی قدم برداشتن تصادفی را نشان می‌دهد.



شکل ۱۵: اجرای نمونه‌ای از مسئله‌ی پیش‌بینی قدم‌برداشتن تصادفی

در صورتی که نوع مسئله را «مسئله‌ی قدم برداشتن تصادفی به روش یادگیری تقویتی» یا RL Random Walk انتخاب نمایید، پنل پارامترهای این نوع مسئله فعّال می‌شود. در این قسمت، مسئله‌ی پیش‌بینی قدم برداشتن تصادفی، با رهیافت یادگیری تقویتی حل می‌شود و نموداری از مقادیر پیش‌بینی شده در اپیزودهای منتخب رسم خواهد شد.

گزینه‌ی Gamma مقدار γ یا ضریب تخفیف را مشخّص می‌کند، که در این مثال خاصّ به راحتی می‌توانید مشاهده کنید که چنان‌چه این ضریب نابرابر با یک باشد، پیش‌بینی به درستی انجام نخواهد شد.

گزینه‌ی Alpha مقدار α یا نرخ یادگیری را تعیین می‌کند.

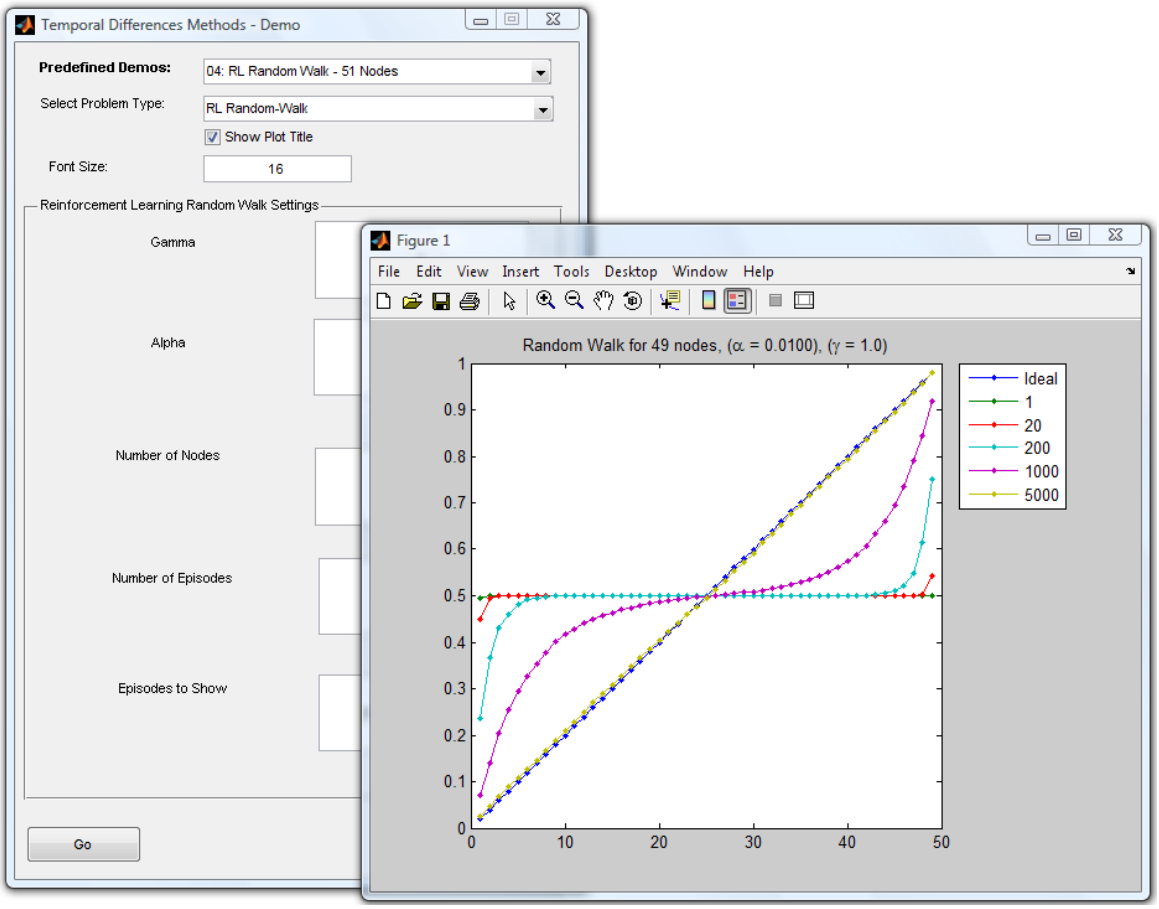
گزینه‌ی Number of Nodes، معرّف تعداد کلّ گره‌های این مسئله است. این تعداد شامل گره پایانی سمت راست و گره پایانی سمت چپ نیز می‌باشد. بنابراین برای حلّ مثالی که در آن تعداد

گره‌های درونی، ۵ گره می‌باشد، باید تعداد کلّ گره‌ها، ۷ وارد شود و در مسئله‌ای که تعداد گره‌های درونی، ۴۹ گره است، برای تعداد کلّ گره‌ها باید، ۵۱، وارد شود.

گزینه‌ی Number of Episodes تعداد اپیزودهایی که به عامل یادگیرنده تغذیه می‌شود برای یادگیری را مشخّص می‌کند. در آغاز هر اپیزود بردار مقادیر V اولّیه، بردار V محاسبه شده در اپیزود قبلی است.

گزینه‌ی Episodes to Show مجموعه‌ی اپیزودهایی را به ازای آن‌ها وضعیت عمل یادگیری عامل در نمودار باید ترسیم شود را تعیین می‌کند. چنان‌چه تمایل دارید نمودار متناظر با هر اپیزودی ترسیم شود، شماره‌ی آن اپیزود را به این بردار اضافه نمایید، به طوری‌که در نهایت مقادیر بردار حاصل صعودی باشد.

شکل ۱۶ اجرای نمونه‌ای از مسئله‌ی قدم‌برداشتن تصادفی به کمک یادگیری تقویتی را نشان می‌دهد.



شکل ۱۶: اجرای نمونه‌ای از مسئله‌ی قدم‌برداشتن تصادفی به کمک یادگیری تقویتی

در صورتی که نوع مسئله را «مسائل جهان مشبّک» یا Grid Worlds انتخاب نمایید، پنل

پارامترهای این نوع مسئله فعّال می‌شود. مسائل جهان مشبّک خود به سه دسته تقسیم می‌شوند، که از طریق گزینه‌ی Grid Types قابل انتخاب‌اند:

۱. دنیای مشبّک ساده

۲. دنیای مشبّک طوفانی

۳. راه‌رفتن کنار درّه

از طریق گزینه‌ی Learning Algorithm می‌توانید الگوریتم یادگیری را انتخاب نمایید. الگوریتم‌هایی که برای هر سه مسئله‌ی فوق پیاده سازی شده‌اند عبارتند از (۱) SARSA و (۲) Q-Learning. در تمامی پیاده‌سازی‌های فوق سیاست عامل ϵ-حریصانه می‌باشد.

از طریق گزینه‌ی Agent Moves، می‌توانید نحوه‌ی حرکت عامل در محیط را تعیین کنید. گزینه‌های قابل انتخاب عبارتند از:

۱. حرکت در چهار جهت اصلی (شمال-جنوب-شرق-غرب)

۲. حرکت شاه (حرکت اریب علاوه بر چهار جهت اصلی)

۳. حرکت شاه به انضمام توقف (علاوه بر حرکت شاه، عامل می‌تواند انتخاب کند که سرجایش بایستد)

در نمایش محیط، هر یک از حرکات با پیکانی در جهت متناظر نمایش داده می‌شود، و عمل توقّف با حرف o نشان داده می‌شود.

گزینه‌ی Gamma مقدار γ یا ضریب تخفیف را مشخّص می‌کند.

گزینه‌ی Alpha مقدار α یا نرخ یادگیری را تعیین می‌کند.

گزینه‌ی Epsilon مقدار ϵ را برای سیاست ϵ-حریصانه تعیین می‌کند.

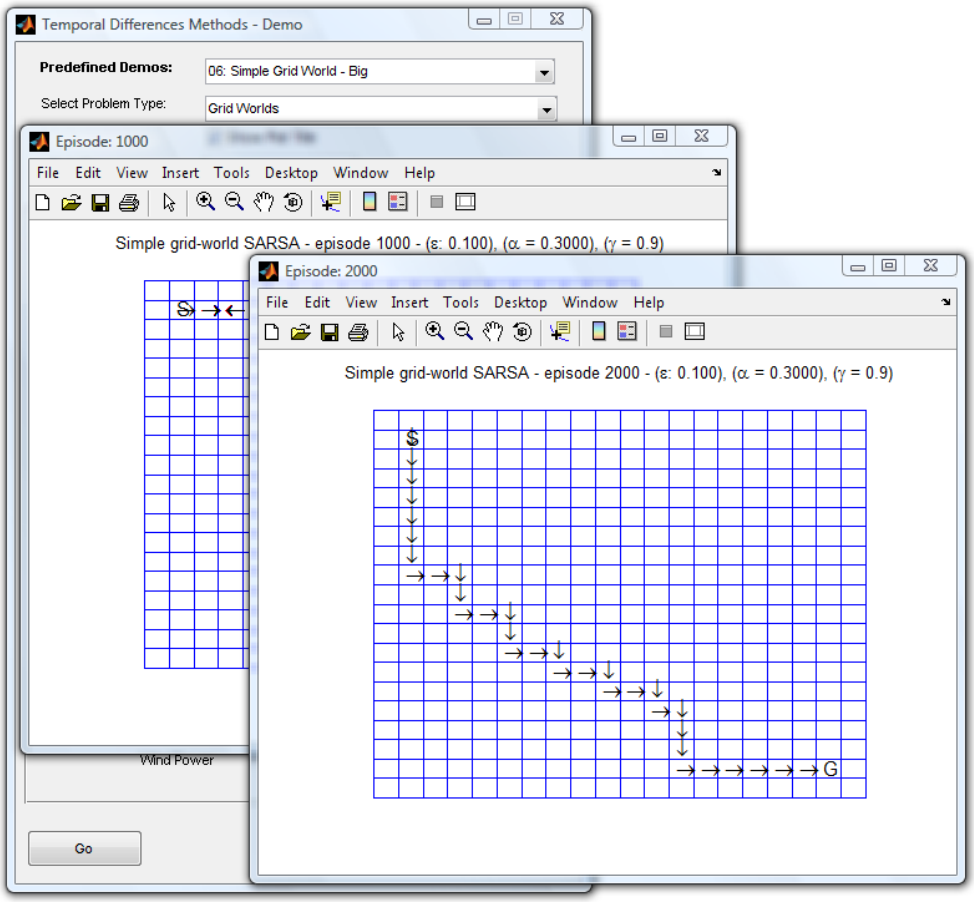
گزینه‌ی Number of Episodes تعداد اپیزودهایی که به عامل یادگیرنده در محیط عمل یادگیری را تجربه می‌کند را مشخّص می‌کند. در آغاز هر اپیزود بردار مقادیر Q اولّیه، بردار Q محاسبه شده در اپیزود قبلی است.

گزینه‌ی Episodes to Show مجموعه‌ی اپیزودهایی را به ازای آن‌ها وضعیت عمل یادگیری عامل نمایش داده می‌شود را تعیین می‌کند. چنان‌چه تمایل دارید شکل متناظر با هر اپیزودی ترسیم شود، شماره‌ی آن اپیزود را به این بردار اضافه نمایید، به طوری‌که در نهایت مقادیر بردار حاصل صعودی باشد.

گزینه‌ی Grid Dimensions ابعاد جهان مشبّک را مشخّص می‌کند. عدد اوّل تعداد سطرها و عدد دوّم تعداد ستون‌ها می‌باشد.

گزینه‌ی Start مختصات نقطه‌ی آغاز را مشخّص می‌کند. عدد اوّل شماره‌ی سطر و عدد دوّم شماره‌ی ستون می‌باشد. در نمایش، این نقطه با حرف S نمایش داده می‌شود.

گزینه‌ی Goal مختصات نقطه‌ی پایان را مشخّص می‌کند. عدد اوّل شماره‌ی سطر و عدد دوّم شماره‌ی ستون می‌باشد. در نمایش، این نقطه با حرف G نمایش داده می‌شود.



شکل ۱۷: اجرای یکی از مسائل جهان مشبّک

گزینه‌ی Wind Powers شدّت باد را در مسئله‌ی جهان مشبّک طوفانی مشخّص می‌کند. مقادیر این بردار متناظرند با ستون‌های جهان مشبّک. بنابراین لازم است که تعداد اعضای این بردار برابر باشند با تعداد ستون‌های جهان مشبّک. هم‌چنین قبل از این‌که برنامه را با مقادیر شدّت باد دلخواه خود اجرا کنید، از این که این مسئله جوابی دارد یا خیر، اطمینان حاصل نمایید. اگر مسئله به ازای مقادیر شدّت بادی که وارد کرده‌اید فاقد جواب باشد، عامل در حلقه‌ی بی‌پایان گرفتار خواهد شد، چرا که همواره برای رسیدن به هدف تلاش می‌کند، در حالی که راهی وجود ندارد و لذا حتّی اپیزود اوّل هم به پایان نخواهد رسید.

شکل ۱۷ اجرای یکی از مسائل جهان مشبّک را نشان می‌دهد.

۱.۴ فایل‌های برنامه

در این بخش فایل‌های برنامه معرفی می‌شوند.

نام فایل	توضیحات
DemoGUI.m	رابط گرافیکی اصلی برنامه
GenerateRandomWalkSequence.m	تولید دنباله‌ای از وضعیت‌های قدم برداشتن تصادفی
PredictionRandomWalk.m	حلّ مسئله‌ی پیش‌بینی قدم برداشتن تصادفی برای λهای مختلف
PredictionRandomWalkAlphaEffect.m	بررسی همزمان مقادیر λ و α برروی دقّت پیش‌بینی
RLRandomWalk.m	حلّ مسئله‌ی قدم برداشتن تصادفی با استفاده از یادگیری تقویتی
DrawGrid.m	رسم یک جدول مشبّک
FindCellCenter.m	یافتن مختصّات مرکز هر یک از خانه‌های جدول
FindColBaseCenter.m	یافتن مختصّات مرکز پایه‌ی هر یک از ستون‌های جدول
DrawTextOnCell.m	نوشتن متن بر روی هر یک از خانه‌های جدول
DrawActionOnCell.m	رسم پیکان‌های متناظر با کنش‌های عامل بر روی هر یک از خانه‌های جدول
DrawEpisodeState.m	رسم هر یک از اپیزودهای مسئله‌ی جهان مشبّک ساده
DrawWindyEpisodeState.m	رسم هر یک از اپیزودهای مسئله‌ی جهان مشبّک طوفانی
DrawCliffEpisodeState.m	رسم هر یک از اپیزودهای مسئله‌ی راه رفتن کنار درّه
GridWorldSARSA.m	حلّ مسئله‌ی جهان مشبّک ساده با الگوریتم SARSA
GridWorldQLearning.m	حلّ مسئله‌ی جهان مشبّک ساده با الگوریتم Q-Learning
WindyGridWorldSARSA.m	حلّ مسئله‌ی جهان مشبّک طوفانی با الگوریتم SARSA
WindyGridWorldQLearning.m	حلّ مسئله‌ی جهان مشبّک طوفانی با الگوریتم Q-Learning
CliffWalkingSARSA.m	حلّ مسئله‌ی راه رفتن کنار درّه با الگوریتم SARSA
CliffWalkingQLearning.m	حلّ مسئله‌ی راه رفتن کنار درّه با الگوریتم Q-Learning

مراجع

[1] Sutton, R. S. and Barto, A. G., "Reinforcement learning: An introduction," 1998.

[2] Sutton, R. S., "Learning to predict by the methods of temporal differences," In *Machine Learning*, pp.9–44, 1988.

[3] Widrow, B. and Hoff, M. E., "Adaptive switching circuits," In *WESCON Convention Record Part IV*, pp.96–104, 1960.

[4] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning internal representations by error propagation," In *Institute for Cognitive Science Technical Report 8506*, La Jolla Ca: University of California, San Diego, 1985.

[5] Mitchell, T. M., "Machine learning," 1997.

[6] Kaelbling, L. P., Littman, M. L., and Moore, A. W., "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, Vol.4, pp.237–285, 1996.

[7] Barto, A. G., "Temporal difference learning," 2007. `http://www.scholarpedia.org/article/Temporal_difference_learning`.