یادگیری تقویتی در محیط‌های پیوسته

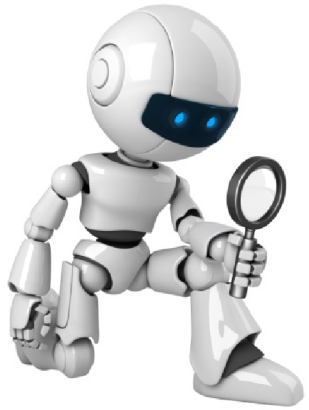
درچند کاربرد

داریوش حسن‌پور

پاییز ۹۴

سرفصل‌ها

- تعریف مساله
- انگیزه؟
- چند کاربرد و راه‌حل
  - دریبل ربات‌ها در مسابقات Robocop
  - یادگیری قایق‌رانی
  - مزایا و معایب روش‌ها
- نتیجه‌گیری
- مراجع



تعریف مساله

- کلیه‌ی الگوریتم‌های یادگیری تقویتی بر اساس «موقعیت، اعمال، پاداش» می‌باشند.

- محیط‌های پیوسته:
  - هریک از «موقعیت» یا «اعمال» به صورت پیوسته تعریف شوند.

- مثال:
  - یادگیری پرواز بالگرد:

    موقعیت: موقعیت جغرافیایی، موقعیت زاویه‌ای، سرعت‌های خطی، سرعت‌های زاویه‌ای

    اعمال: سرعت‌های زاویه‌ای پره‌ها، تنظیم پارامترهای دستگاه‌های کنترلی، ....
  - یادگیری رانندگی:

    موقعیت: موقعیت جغرافیایی، زاویه‌ی فرمان، میزان گاز، میزان ترمز، فاصله از اطراف و ...

    اعمال: میزان گاز، میزان ترمز، میزان کلاچ، زاویه‌ی فرمان و ...

انگیزه؟

- مشکل کاربردهای دنیای واقعی با یادگیری تقویتی چیست؟
  ۱. برای یادگیری یک سیاست حداقل باید چندین هزار دور ربات به اجرا دربیاید.
  ۲. علاوه بر زمان یادیگری، زمان تنظیم و کالیبره کردن سنسورها، صدمات احتمالی به ربات.

- مشکل روش‌های موجود با محیطهای پیوسته چیست؟
  ۱. حافظه
  ۲. زمان
  ۳. عدم تضمین همگرایی

انگیزه؟

- راه‌حل‌های احتمالی برای یادگیری تقویتی در محیط‌های پیوسته:
  - گسسته‌سازی
  - مدل‌سازی و یادگیری مدل
  - کاهش ابعاد
  - یادگیری توزیع‌شده(ماژولار)

چند کاربرد و راه‌حل

درییل ربات‌ها در مسابقات ROBOCOP

دریبل ربات‌ها در مسابقات Robocop

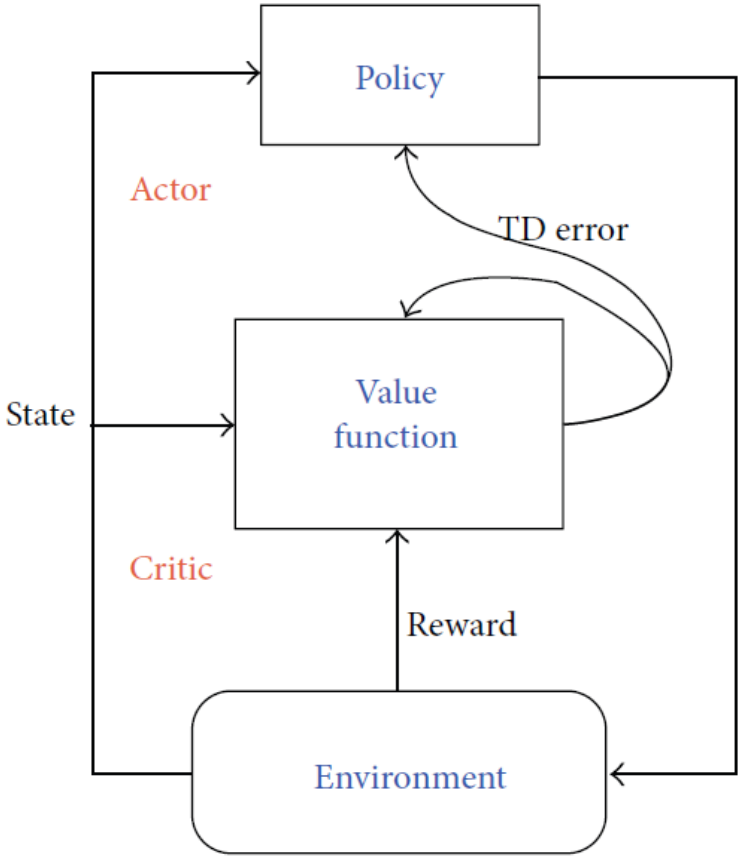
- در این مقاله یک معماری‌ای معرفی شده است که برای یادگیری دریبل ربات‌ها برای مسابقات Robocop استفاده شده است.

- تعریف مساله:
  - موقعیت: به تعداد نامتناهی و پیوسته که توسط یک بردار از اعداد پیوسته نمایش داده می‌شود.
  - اعمال: به تعداد متناهی به ازای هر موقعیت، هرعمل یک بردار از اعداد حقیقی از پارامتر می‌باشد.

- هدف:
  - توزیع یادگیری سیاست بهینه کلی به دو عامل یادگیری:
    ۱. به ازای هر موقعیتی چه عملی انجام شود؟
    ۲. پارامترهای عمل انتخابی چگونه باشند؟

درییل ربات‌ها در مسابقات Robocop

- معماری رایج برای یادگیری تقویتی

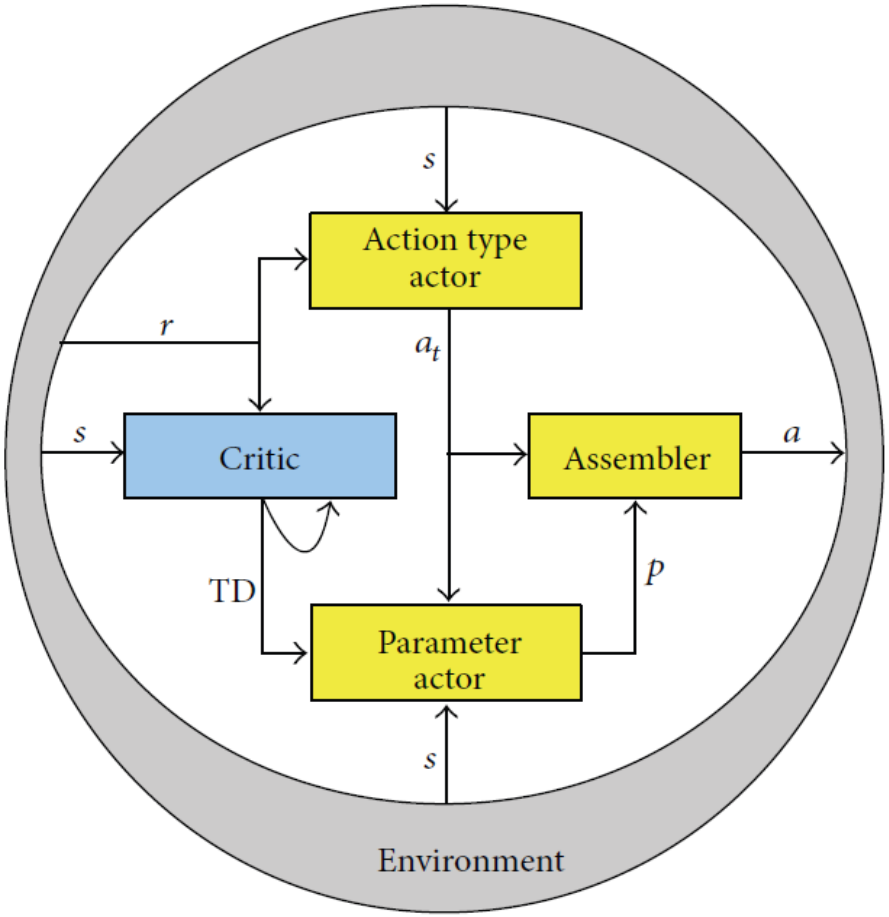


$$V_{t+1}(s) \longleftarrow V_t(s) + \alpha\left[r + \gamma V_t\left(s'\right) - V_t(s)\right]$$

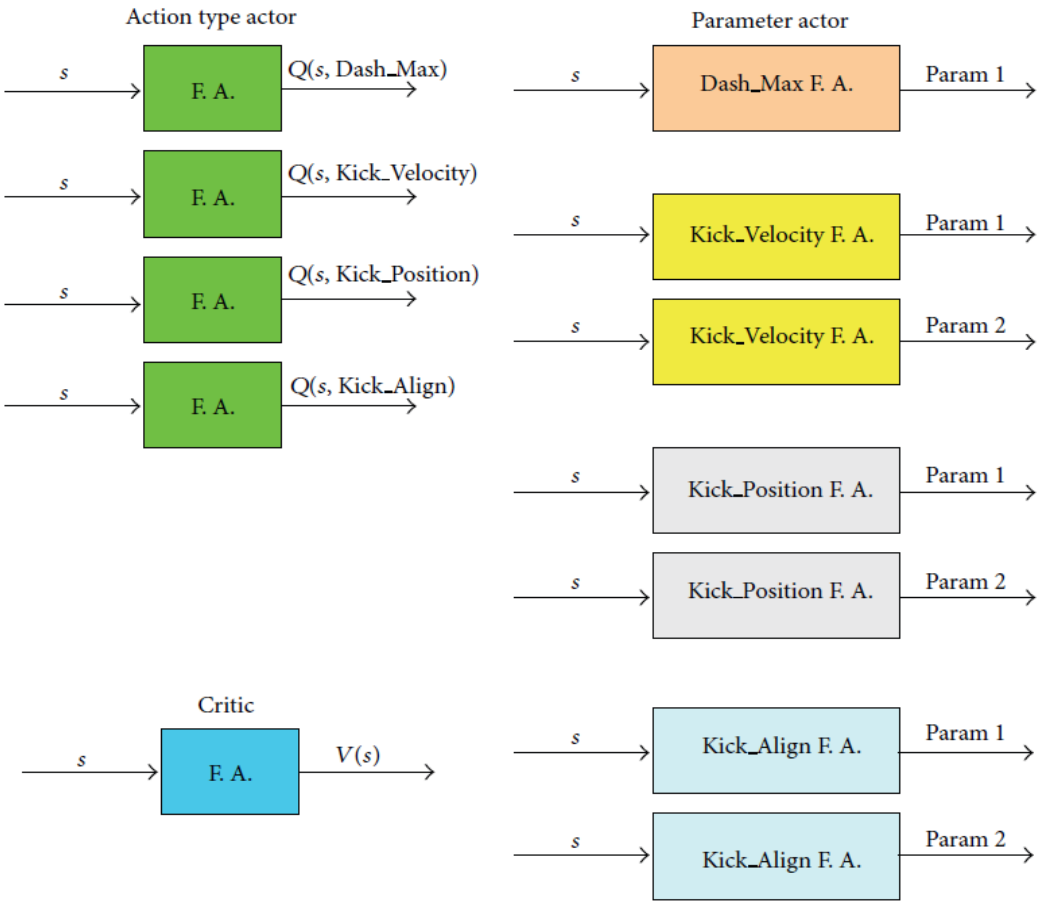
$$\delta_t = r_{t+1} + \gamma V\left(s_{t+1}\right) - V\left(s_t\right)$$

درییل ربات‌ها در مسابقات Robocop

- معماری معرفی شده:



درریبل ربات‌ها در مسابقات Robocop



درییل ربات‌ها در مسابقات Robocop

(1) Initialize the *action type actor*, the *parameter actor,* and the *critic*. This step refers to randomly choosing the initial values of all the parameters used by all the function approximators.

(2) From current state s, select the best action type a_t and parameter vector $\vec{p}_{a_t}$. To select the best action type, we simply evaluate all the function approximators used by the action type actor, with the current state s, and we pick the action type whose function approximator gives the greatest evaluation. Once we have selected the best action type for the current state s, we evaluate the function approximators assigned to that action type, to get $\vec{p}_{a_t}$.

(3) Assembly action a with action type a_t and parameter vector $\vec{p}_{a_t}$. This step is a plain call to the code function used internally by our agent to get ready to execute the chosen action.

(4) For each training episode, use learning rate α and discount factor γ to do the following.

(a) Execute action a and observe next state s' and the scalar reward r.

(b) Compute the TD error as $\varepsilon = [r + \gamma \widehat{V}(s')] - \widehat{V}(s)$, where $\widehat{V}(s)$ is the state value function stored by the critic.

(c) Update the critic using the TD error with $\widehat{V}(s) \leftarrow \widehat{V}(s) + \alpha\varepsilon$.

(d) If $(\varepsilon > 0)$ then reinforce the use of $\vec{p}_{a_t}$ by retraining the function approximator of action type a_t with the example $(s, \vec{p}_{a_t})$.

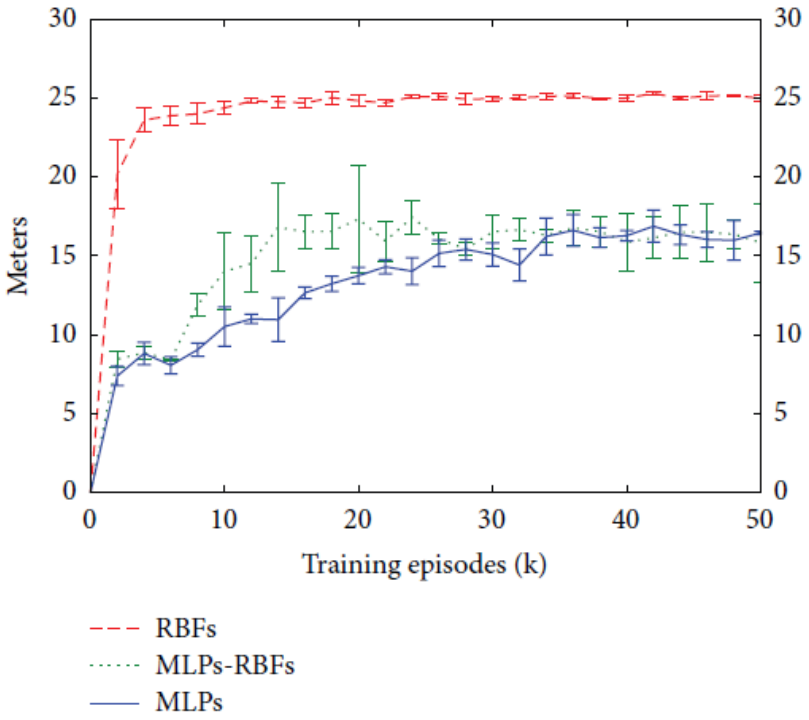
(e) From the next state s', compute the next action type a_t' and parameter vector $\vec{p}_{a_t'}$.

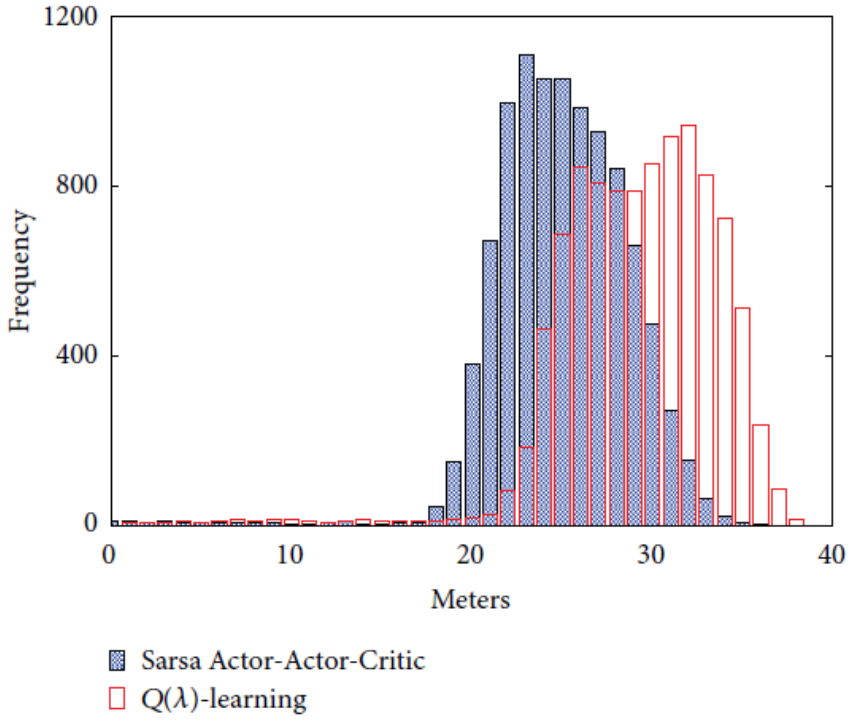
(f) Assemble the next action a' with action type a_t' and parameter vector $\vec{p}_{a_t'}$.

(g) Update the action type actor with $Q(s, a_t) \leftarrow Q(s, a_t) + \alpha[r + \gamma Q(s', a_t') - Q(s, a_t)]$, where $Q(s, a_t)$ is the state-action value function implemented as a function approximator.

(h) Update the current state with $s \leftarrow s'$ and the current action with $a \leftarrow a'$.

درییل ربات‌ها در مسابقات Robocop





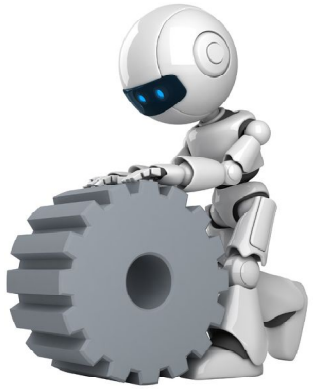
دریبل ربات‌ها در مسابقات Robocop

Table 1: Comparison of the best policies for the dribbling problem.

	SARSA A^2C	$Q(\lambda)$-learning
Algorithm type	Actor-Critic	$Q(\lambda)$-learning
Function approx.	RBFs	CMACs
States	Continuous	Continuous
Actions	Continuous	Discrete
Total learning time	10 minutes	24 hours 30 minutes
Average distance	25.45 meters	29.21 meters
Maximum distance	36.23 meters	39.0 meters

چند کاربرد و راه‌حل

یادگیری قایق‌رانی



یادگیری قایق‌رانی

$$\mathcal{A}(s) = \{a_1, a_2, \cdots, a_N\}, \quad a_i \sim \pi^0(a|s).$$

$$\pi^0(a|s) \simeq \sum_{i=1}^{N} w_i \cdot \delta(a - a_i), \quad \delta_a(x) = \frac{1}{a\sqrt{\pi}} e^{-x^2/a^2} \quad a_i \in \mathcal{A}(s),\ w_i \in \mathcal{W}(s)$$

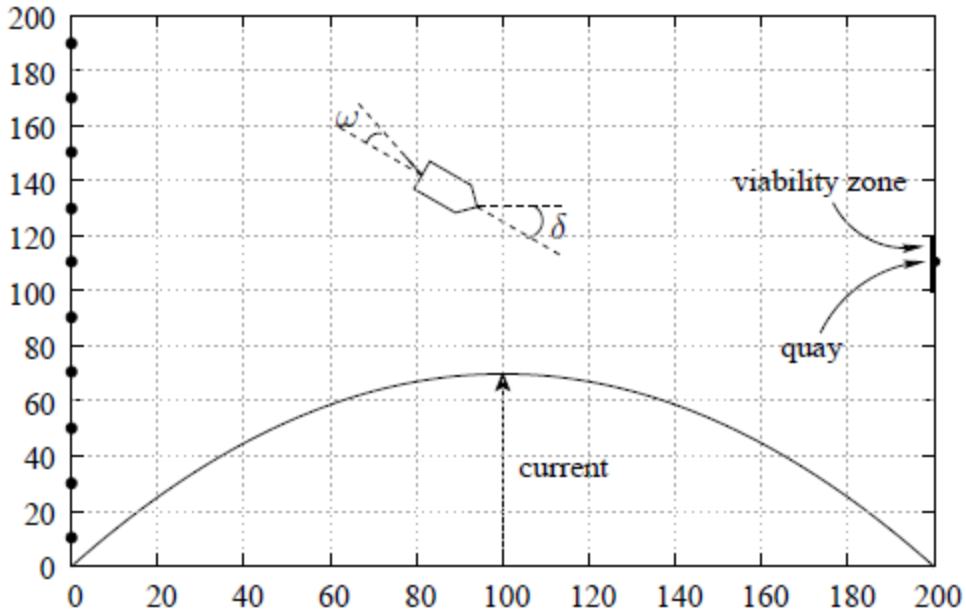
یادگیری قایق‌رانی

Algorithm 1 SMC-learning algorithm

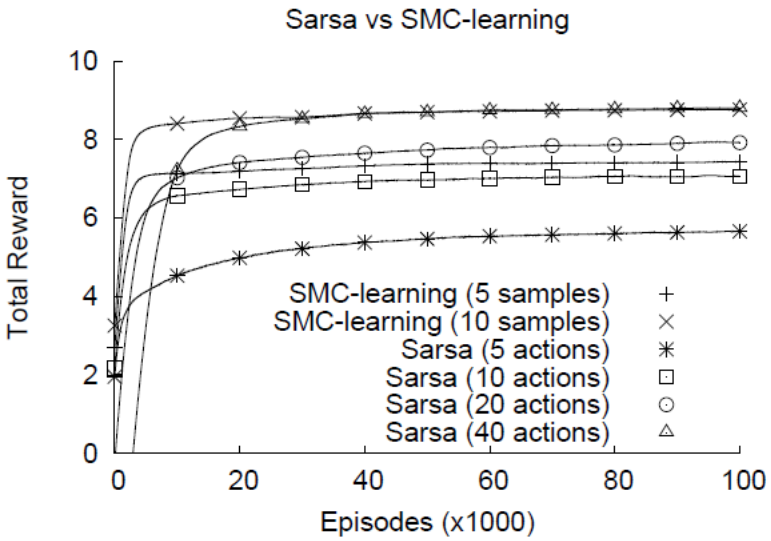
for all $s \in \mathcal{S}$ **do**
  Initialize $\mathcal{A}(s)$ by drawing N samples from $\pi^0(a|s)$
  Initialize $\mathcal{W}(s)$ with uniform values: $w_i = 1/N$
end for
for each time step t **do**
  Action Selection
  Given the current state s_t, the actor selects action a_t from $\mathcal{A}(s_t)$ according to $\pi^t(a|s) = \sum_{i=1}^{N} w_i \cdot \delta(a - a_i)$
  Critic Update
  Given the reward r_t and the utility of next state s_{t+1}, the critic updates the action value $Q(s_t, a_t)$
  Actor Update
  Given the action-value function, the actor updates the importance weights
  if the weights have a high variance **then**
    the set $\mathcal{A}(s_t)$ is resampled
  end if
end for

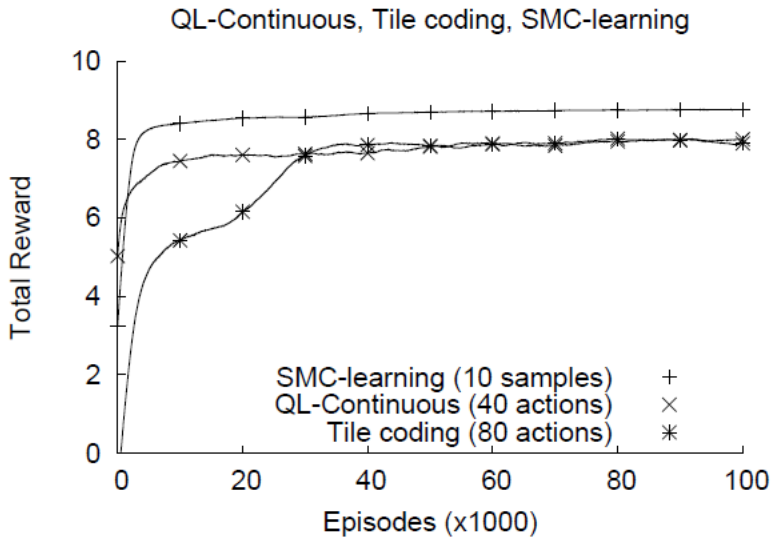
$$w_i^{t+1} = w_i^t \frac{e^{\frac{\Delta Q^{t+1}(s,a_i)}{\tau}}}{\sum_{j=1}^{N} w_j e^{\frac{\Delta Q^{t+1}(s,a_j)}{\tau}}}, \quad \Delta Q^{t+1}(s,a_i) = Q^{t+1}(s,a_i) - Q^t(s,a_i)$$

یادگیری قایق‌رانی



یادگیری قایق‌رانی





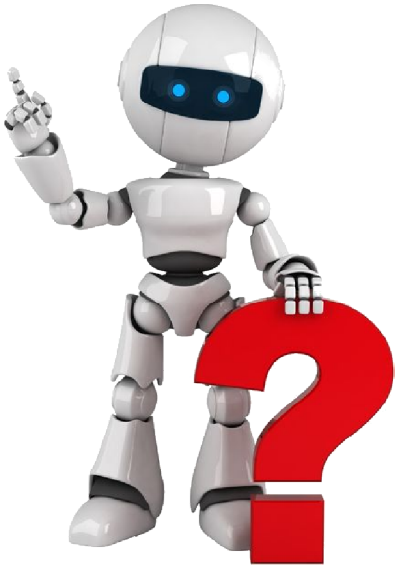
مزایا و معایب روشهای معرفی شده؟

- مزایا:
  - ساده.
  - میتوانند سیاست بهینه را برای موقعیت‌های نامتناهی با اعمال متناهی پیوسته را یادبگیرند.
- معایب:
  - همگرایی کند.
  - نمی‌توانند برای مسایلی باموقعیت‌ها و اعمال نامتنهای و پیوسته بکار روند.

نتیجه‌گیری

- برخی از نقاط ضعف یادگیری تقویتی معرفی شد.
- دو نمونه از مثالهای یادگیری در محیطهای پیوسته معرفی شدند.
- یک معماری برای یادگیری در محیطهای پیوسته و اعمال محدود معرفی شد.
- یک الگوریتم برای یادگیری میزان اهمیت و انتخاب اعمال پیوسته و محدود ارایه شد.

با تشکر



مراجع

1. Uc-Cetina, Víctor. "A novel reinforcement learning architecture for continuous state and action spaces." *Advances in Artificial Intelligence* 2013 (2013): 7.

2. Lazaric, Alessandro, Marcello Restelli, and Andrea Bonarini. "Reinforcement learning in continuous action spaces through sequential Monte Carlo methods." *Advances in neural information processing systems*. 2007.