“One learns from books and example only that certain things can be done. Actual learning requires that you do those things.”

Frank Herbert, *Children of Dune*

CMPUT 365
Introduction to RL

Marlos C. Machado

https://www.duneawakening.com/en/developer-update-2-sounds-of-arrakis

Classes 30-32/36

Coursera Reminder

You **should be enrolled in the private session** we created in Coursera for CMPUT 365.

I **cannot** use marks from the public repository for your course marks.

You **need** to **check**, **every time**, if you are in the private session and if you are submitting quizzes and assignments to the private section.

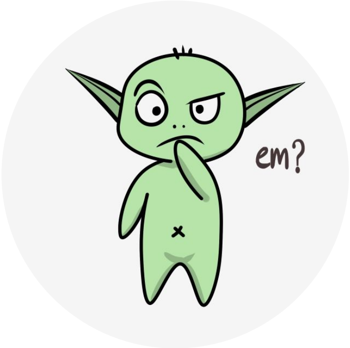
The deadlines in the public session **do not align** with the deadlines in Coursera.

If you have any questions or concerns, **talk with the TAs** or email us `cmput365@ualberta.ca`.

Reminders and Notes

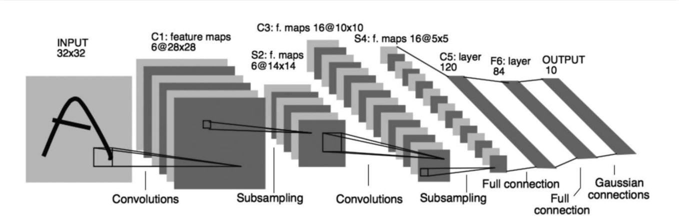
- The practice quiz and programming assignment for “Control with FA” are due on Friday (only 2 to go!).
- The Student Perspectives of Teaching (SPOT) Survey is now available.
  `https://go.blueja.io/qVavvKlQuUedfy1YlM1_kA`

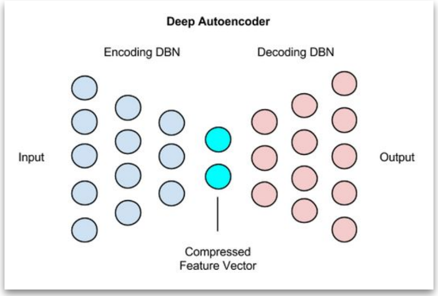
Please, interrupt me at any time!



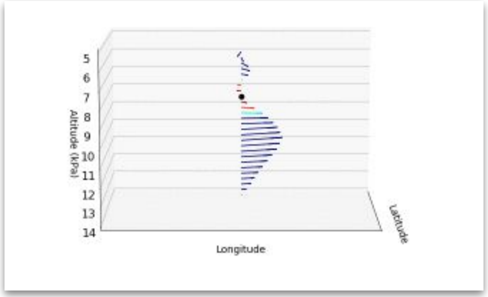
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Last Class: Neural Networks and Feature/Input Construction





https://wiki.pathmind.com/deep-autoencoder



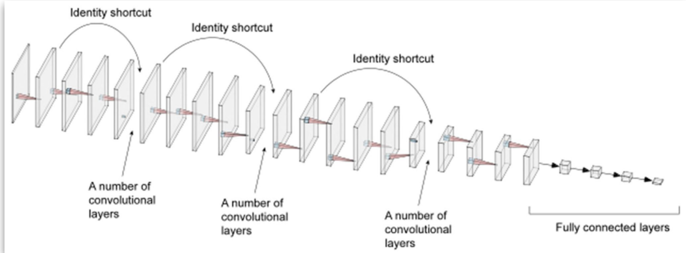
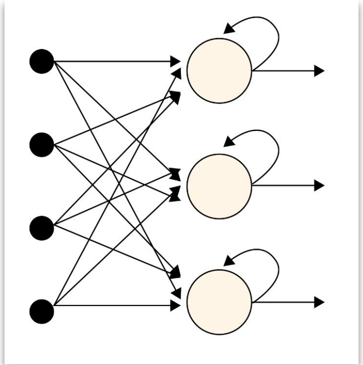


Image by Yu, Miao, and Wang (2022)



https://www.scaler.com/topics/deep-learning/rnn/

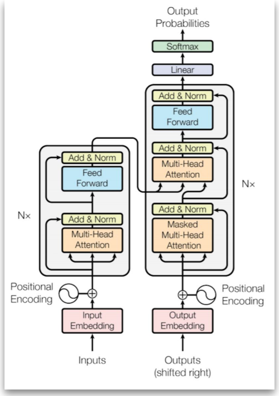
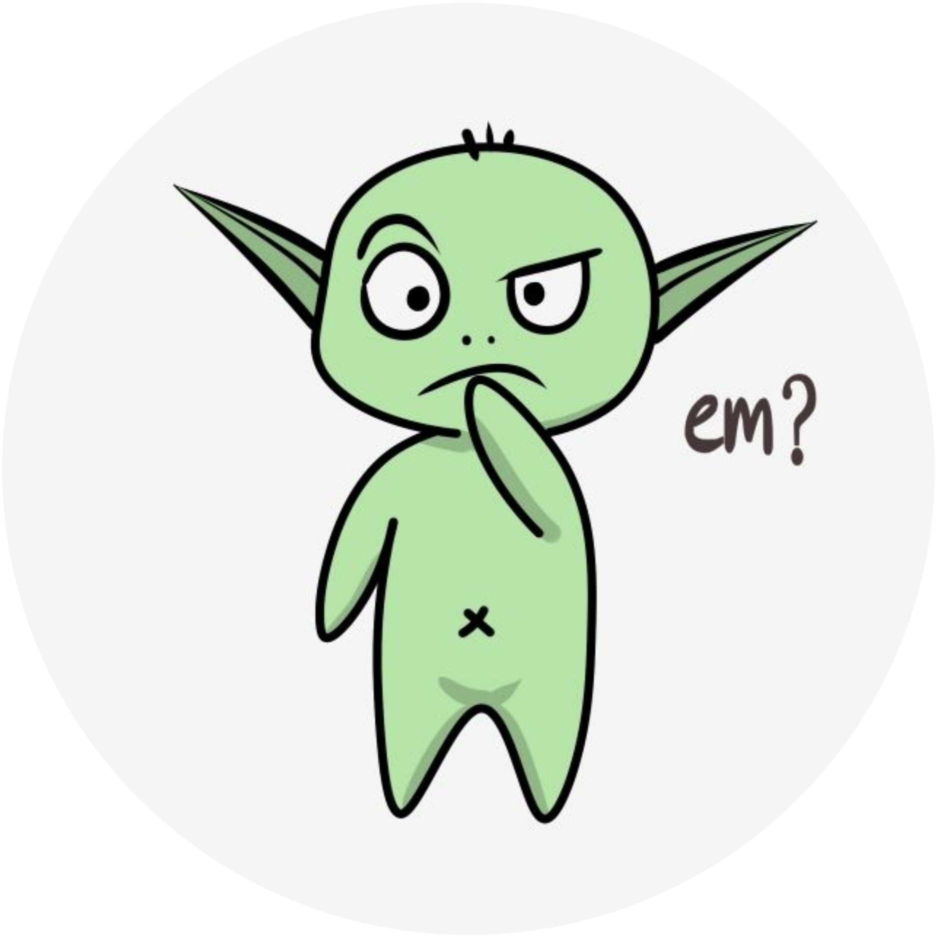


Image by Vaswani et al. (2017)



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Episodic Semi-gradient Control

- We need to approximate the action-value function now, $\hat{q} \approx q_\pi$, that is represented as a parameterized function form with weight vector **w**.
- Before (until last class): $S_t \mapsto U_t$.
  Now: $S_t, A_t \mapsto U_t$.

Episodic Semi-gradient Control

- We need to approximate the action-value function now, $\hat{q} \approx q_\pi$, that is represented as a parameterized function form with weight vector **w**.
- Before (until last class): $S_t \mapsto U_t$.
  Now: $S_t, A_t \mapsto U_t$.
- Action-value prediction:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \Big[U_t - \hat{q}(S_t, A_t, \mathbf{w}_t) \Big] \nabla \hat{q}(S_t, A_t, \mathbf{w}_t)$$

Episodic Semi-gradient Control

- We need to approximate the action-value function now, $\hat{q} \approx q_\pi$, that is represented as a parameterized function form with weight vector **w**.
- Before (until last class): $S_t \mapsto U_t$.
  Now: $S_t, A_t \mapsto U_t$.
- Action-value prediction:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \Big[U_t - \hat{q}(S_t, A_t, \mathbf{w}_t) \Big] \nabla \hat{q}(S_t, A_t, \mathbf{w}_t)$$

- Episodic semi-gradient one-step *Sarsa*:

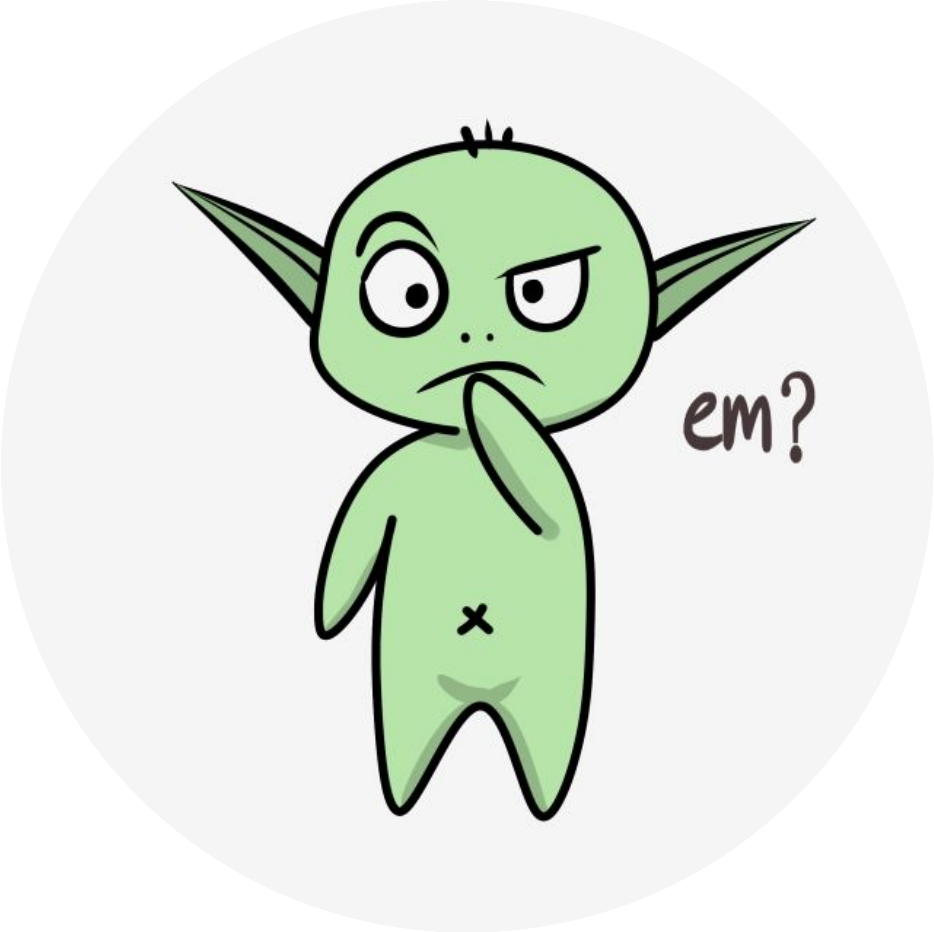
$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \Big[R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t) \Big] \nabla \hat{q}(S_t, A_t, \mathbf{w}_t)$$

Episodic Semi-gradient Sarsa

Episodic Semi-gradient Sarsa for Estimating $\hat{q} \approx q_*$

Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \rightarrow \mathbb{R}$
Algorithm parameters: step size $\alpha > 0$, small $\varepsilon > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
    $S, A \leftarrow$ initial state and action of episode (e.g., ε-greedy)
    Loop for each step of episode:
        Take action A, observe R, S'
        If S' is terminal:
            $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
            Go to next episode
        Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
        $S \leftarrow S'$
        $A \leftarrow A'$

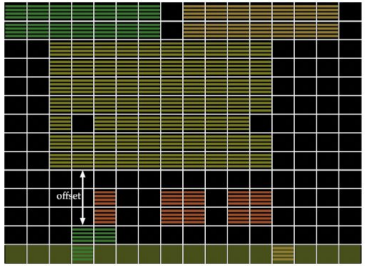


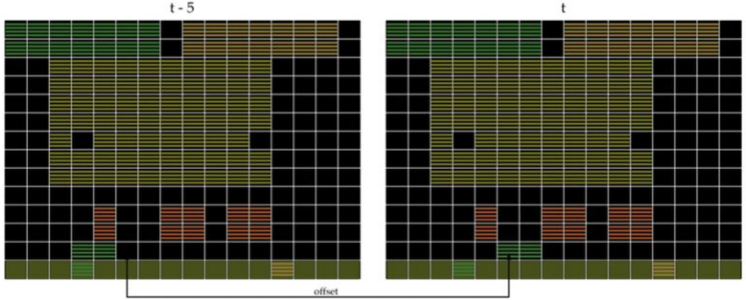
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

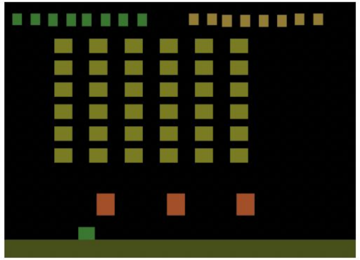
This really works!

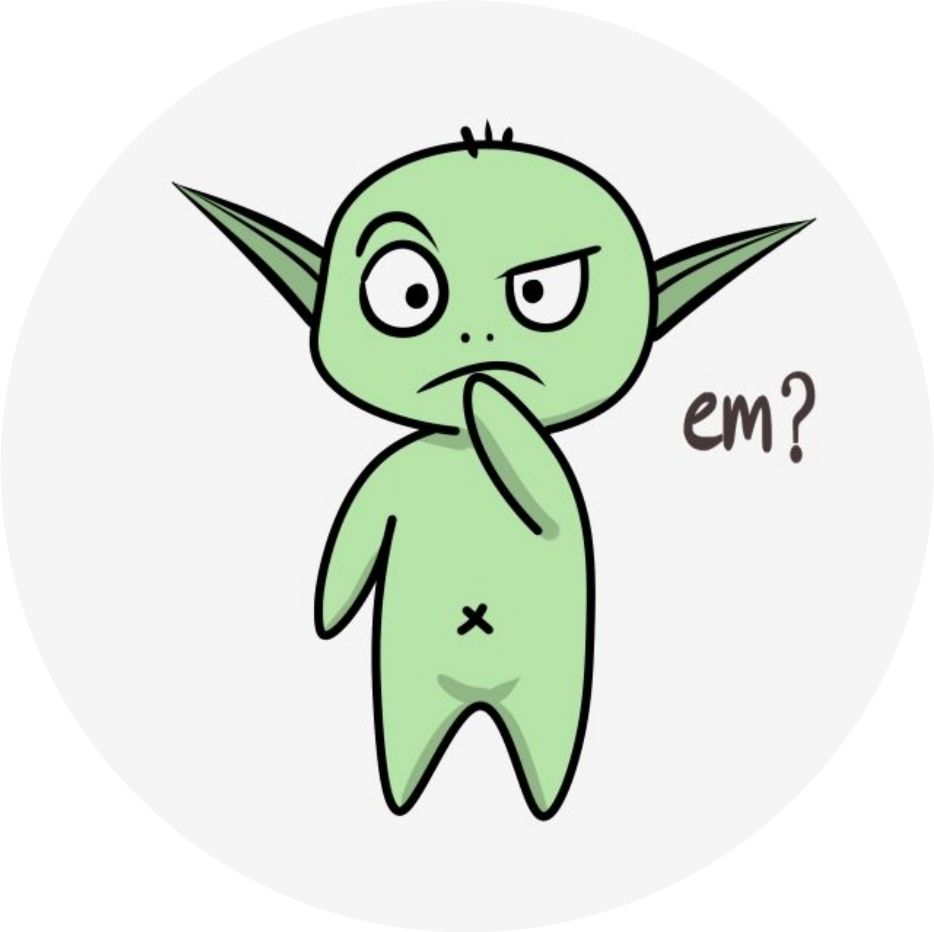
State of the Art Control of Atari Games Using Shallow Reinforcement Learning

Yitao Liang†, Marlos C. Machado‡, Erik Talvitie†, and Michael Bowling‡
†Franklin & Marshall College
Lancaster, PA, USA
{yliang, erik.talvitie}@fandm.edu
‡University of Alberta
Edmonton, AB, Canada
{machado, mbowling}@ualberta.ca



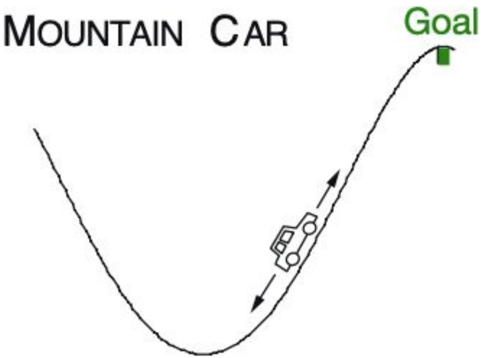






https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Example: Mountain Car Task

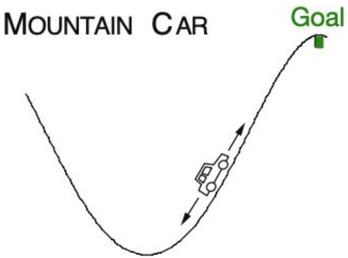


- Observations: $(x, \dot{x})$
- Actions:
  - Full throttle forward: +1
  - Full throttle reverse: -1
  - Zero throttle: 0
- Rewards: -1 at every time step, until end of episode.
- Dynamics:

$$x_{t+1} \doteq bound\big[x_t + \dot{x}_{t+1}\big]$$

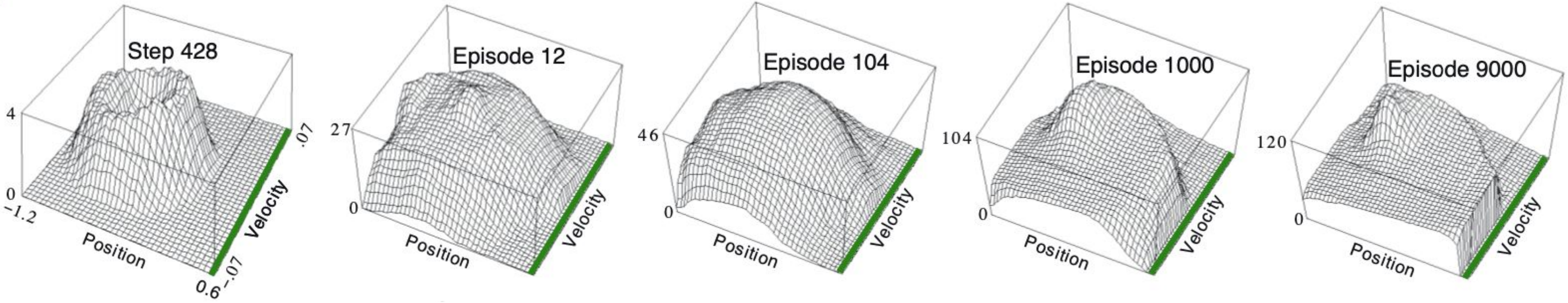
$$\dot{x}_{t+1} \doteq bound\big[\dot{x}_t + 0.001A_t - 0.0025\cos(3x_t)\big]$$

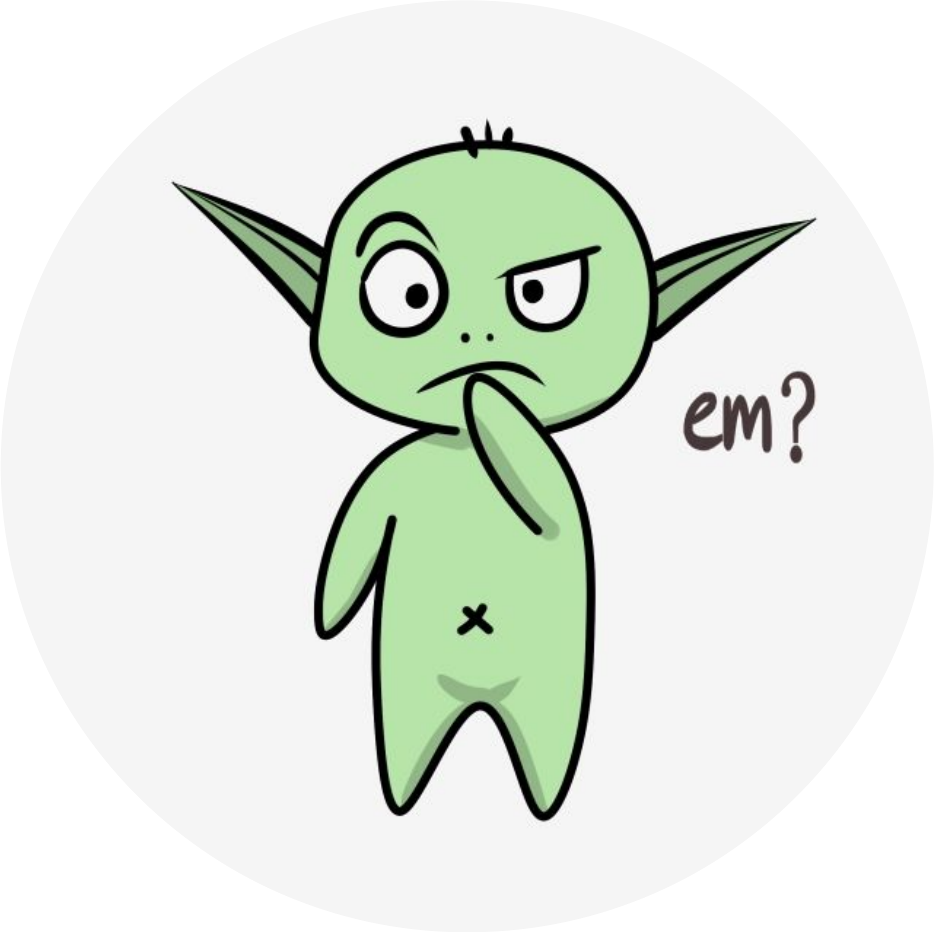
$$-1.2 \le x_{t+1} \le 0.5 \text{ and } -0.07 \le \dot{x}_{t+1} \le 0.07$$



"Solution": Mountain Car Task

- Feature representation:
  - Grid-tilings with 8 tilings and asymmetrical offsets.
  - $\hat{q}(s, a, \mathbf{w}) \doteq \mathbf{w}^\top \boxed{\mathbf{x}(s, a)} = \sum_{i=1}^{d} w_i \cdot x_i(s, a)$
- Sarsa
  - Weights initialized at zero. Effectively optimistic initialization.





https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Journal of Artificial Intelligence Research 47 (2013) 253–279 Submitted 02/13; published 06/13

The Arcade Learning Environment: An Evaluation Platform for General Agents

Marc G. Bellemare MG17@CS.UALBERTA.CA
University of Alberta, Edmonton, Alberta, Canada

Yavar Naddaf YAVAR@EMPIRICALRESULTS.CA
Empirical Results Inc., Vancouver,
British Columbia, Canada

Joel Veness VENESS@CS.UALBERTA.CA
Michael Bowling BOWLING@CS.UALBERTA.CA
University of Alberta, Edmonton, Alberta, Canada

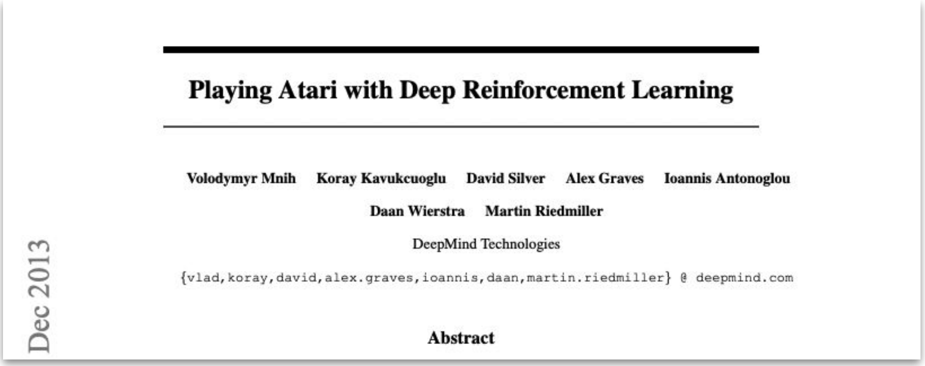
Abstract

In this article we introduce the Arcade Learning Environment (ALE): both a challenge problem and a platform and methodology for evaluating the development of general, domain-independent AI technology. ALE provides an interface to hundreds of Atari 2600 game environments, each one different, interesting, and designed to be a challenge for human players. ALE presents significant research challenges for reinforcement learning, model learning, model-based planning, imitation learning, transfer learning, and intrinsic motivation. Most importantly, it provides a rigorous testbed for evaluating and comparing approaches to these problems. We illustrate the promise of ALE by developing and benchmarking domain-independent agents designed using well-established AI techniques for both reinforcement learning and planning. In doing so, we also propose an evaluation

Arcade Learning Environment

Over 50 Domains in 8 Minutes 23 Seconds

Deep Q-Network (and Deep RL)
[Mnih et al., 2013, 2015]



Dec 2013

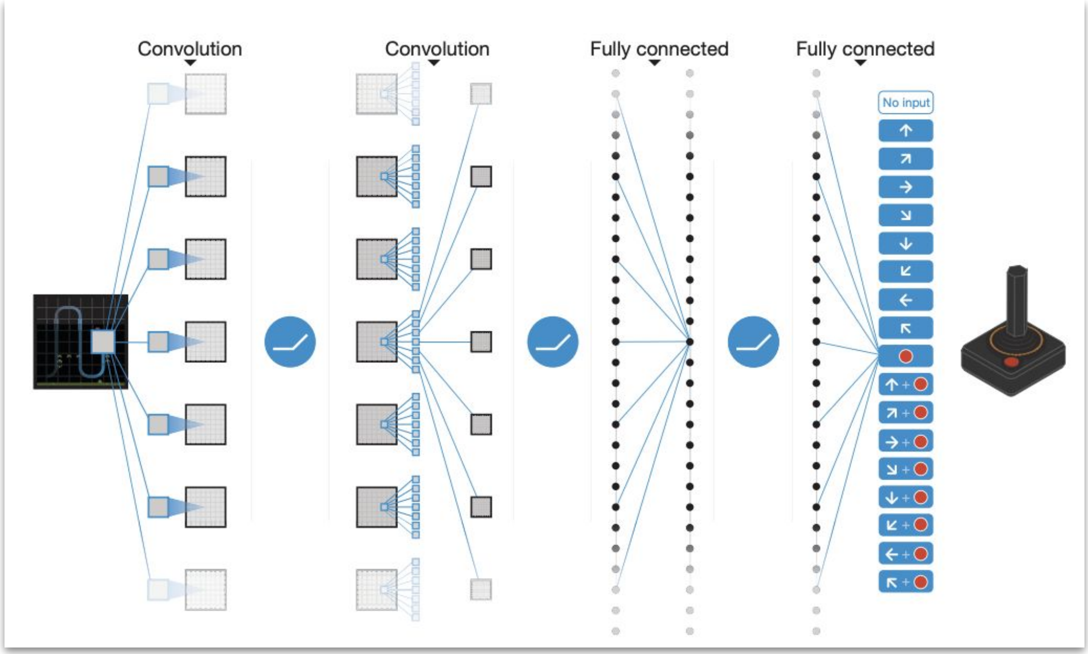
Playing Atari with Deep Reinforcement Learning

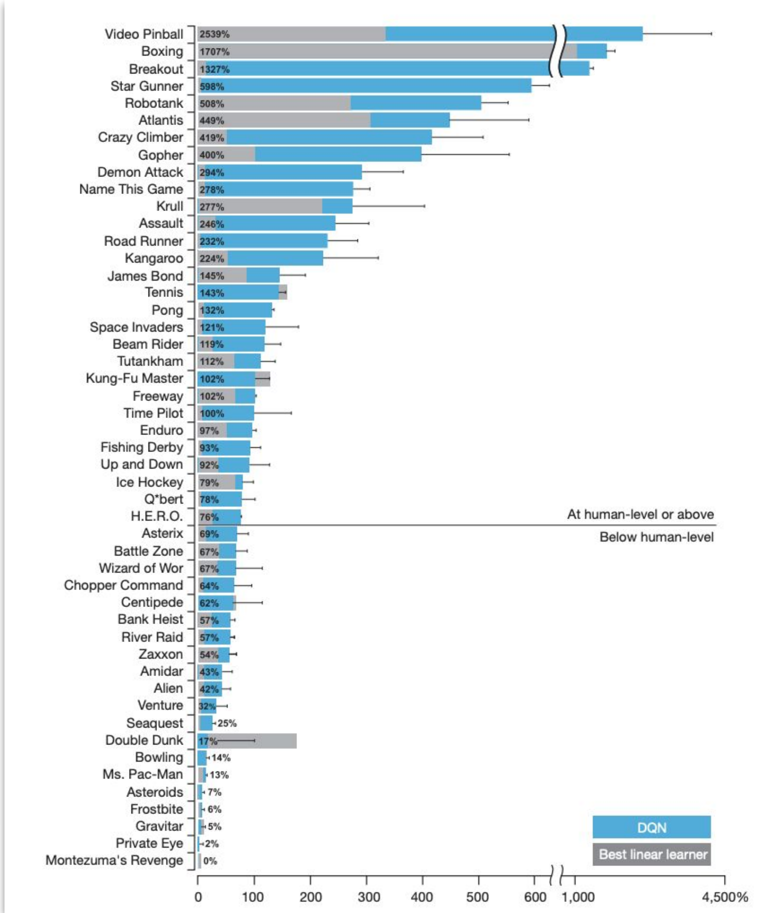
Volodymyr Mnih Koray Kavukcuoglu David Silver Alex Graves Ioannis Antonoglou
Daan Wierstra Martin Riedmiller
DeepMind Technologies
{vlad,koray,david,alex.graves,ioannis,daan,martin.riedmiller} @ deepmind.com

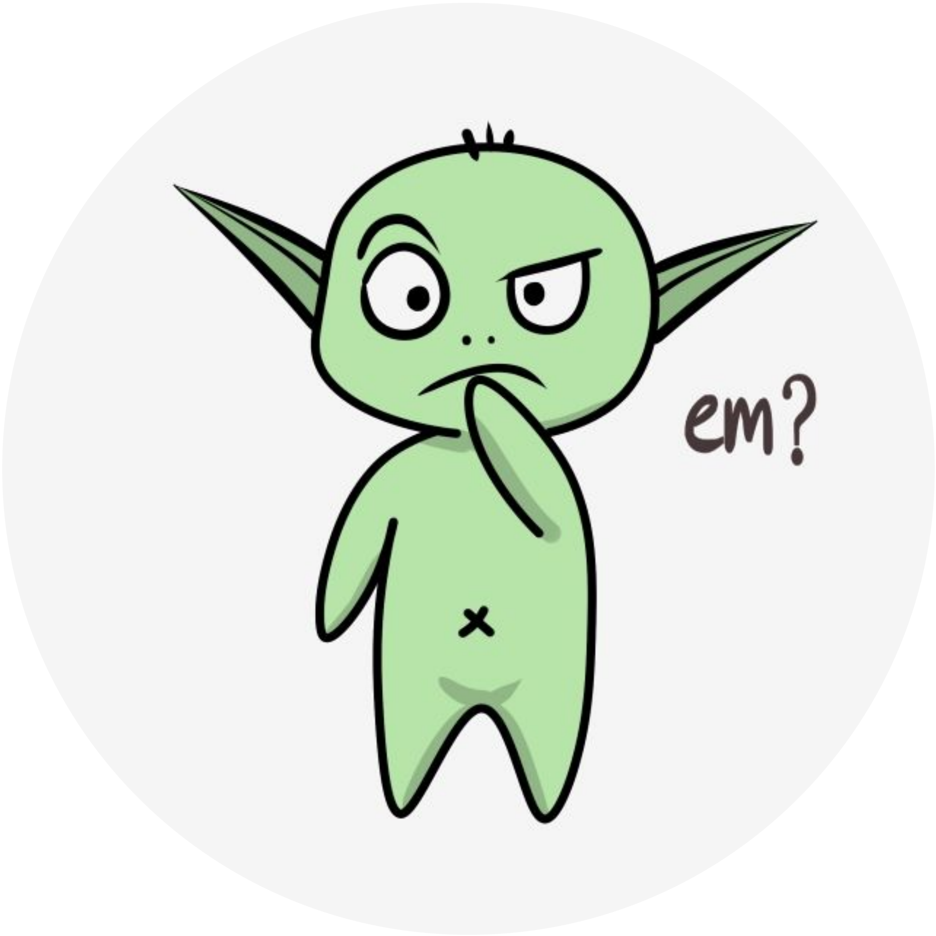
Abstract

Deep Q-Network (and Deep RL)

[Mnih et al., 2013, 2015]







https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Deep Q-Network (DQN)

[Mnih et al., 2013, 2015]

Stacked frames

-1, 0, +1 rewards

Clipped error term

$$\mathcal{L}^{\text{DQN}} = \mathbb{E}_{(s,a,r,s') \sim U(\mathcal{D})}\left[\left(R_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q(S_{t+1}, a'; \boldsymbol{\theta}^{-}) - Q(S_t, A_t, \boldsymbol{\theta}_t)\right)^2\right]$$

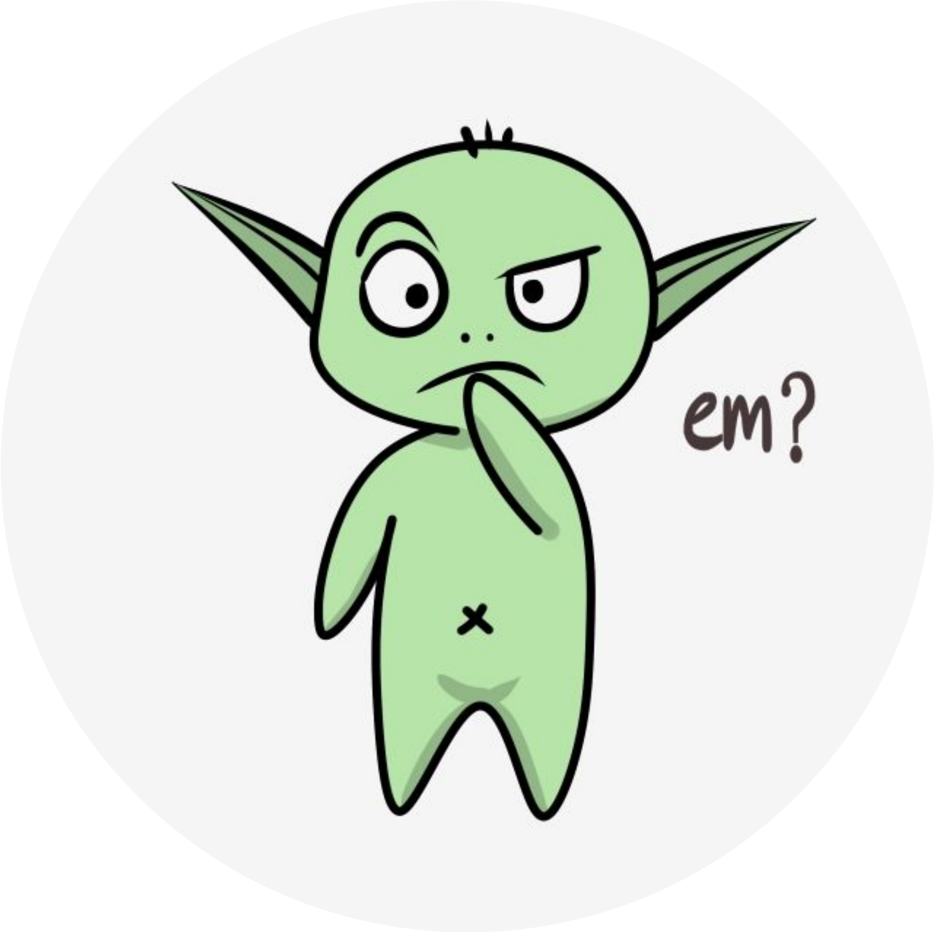
Experience replay buffer (Lin, 1993)
Original size: 1M frames

Target network
Original update frequency: 10k

ε decay
Originally, from 1.0 to 0.1 over 1M frames

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha\left[R_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q(S_{t+1}, a'; \boldsymbol{\theta}^{-}) - Q(S_t, A_t, \boldsymbol{\theta}_t)\right]\nabla_{\boldsymbol{\theta}_t} Q(S_t, A_t; \boldsymbol{\theta}_t)$$

RMSProp



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Deep Q-Network (DQN)

[Mnih et al., 2013, 2015]

Game	Random Play	Best Linear Learner	Contingency (SARSA)	Human	DQN (± std)	Normalized DQN (% Human)
Alien	227.8	939.2	103.2	6875	3069 (±1093)	42.7%
Amidar	5.8	103.4	183.6	1676	739.5 (±3024)	43.9%
Assault	222.4	628	537	1496	3359(±775)	246.2%
Asterix	210	987.3	1332	8503	6012 (±1744)	70.0%
Asteroids	719.1	907.3	89	13157	1629 (±542)	7.3%
Atlantis	12850	62687	852.9	29028	85641(±17600)	449.9%
Bank Heist	14.2	190.8	67.4	734.4	429.7 (±650)	57.7%
Battle Zone	2360	15820	16.2	37800	26300 (±7725)	67.6%
Beam Rider	363.9	929.4	1743	5775	6846 (±1619)	119.8%
Bowling	23.1	43.9	36.4	154.8	42.4 (±88)	14.7%

Deep Q-Network (DQN)

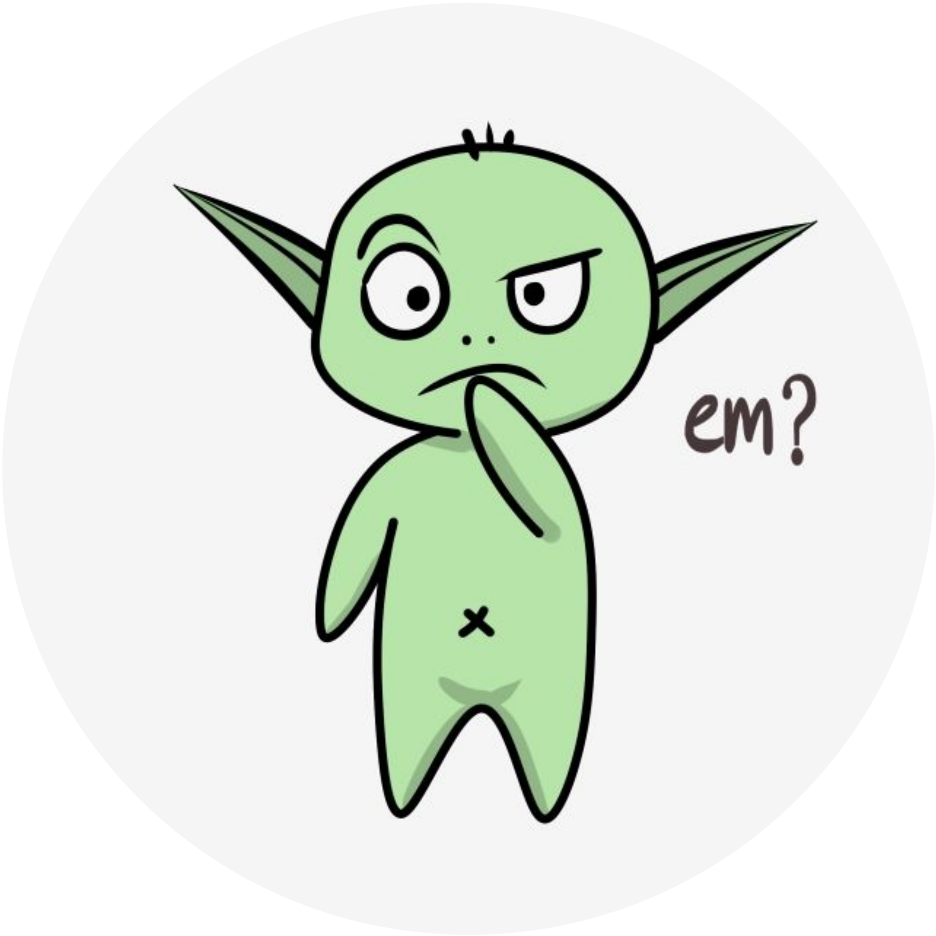
[Mnih et al., 2013, 2015]

Game	Random Play	Best Linear Learner	Contingency (SARSA)	Human	DQN (± std)	Normalized DQN (% Human)
Alien	227.8	939.2	103.2	6875	3069 (±1093)	42.7%
Amidar	5.8	103.4	183.6	1676	739.5 (±3024)	43.9%
Assault	222.4	628	537	1496	3359(±775)	246.2%
Asterix	210	987.3	1332	8503	6012 (±1744)	70.0%
Asteroids	719.1	907.3	89	13157	1629 (±542)	7.3%
Atlantis	12850	62687	852.9	29028	85641(±17600)	449.9%
Bank Heist	14.2	190.8	67.4	734.4	429.7 (±650)	57.7%
Battle Zone	2360	15820	16.2	37800	26300 (±7725)	67.6%
Beam Rider	363.9	929.4	1743	5775	6846 (±1619)	119.8%
Bowling	23.1	43.9	36.4	154.8	42.4 (±88)	14.7%

Tables can be misleading

[Machado et al., 2018]

- Tables imply an apples-to-apples comparison, even when they are not:
- DQN saw much more data than the baselines.
- DQN measured its performance differently than the baselines.
- DQN used domain knowledge other baselines didn’t:
  - Lives signal
  - Action set



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Avg. Reward: A Problem Setting for Continuing Tasks

- Continuing problems without discounting.
  - The agent cares about all rewards equally.

Avg. Reward: A Problem Setting for Continuing Tasks

- Continuing problems without discounting.
    - The agent cares about all rewards equally.
- Quality of a policy is defined by the average rate of reward, r(π):

$$
\begin{aligned}
r(\pi) &\doteq \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h} \mathbb{E}[R_t \mid S_0, A_{0:t-1} \sim \pi] \\
&= \lim_{t\to\infty} \mathbb{E}[R_t \mid S_0, A_{0:t-1} \sim \pi], \\
&= \sum_s \mu_\pi(s) \sum_a \pi(a|s) \sum_{s',r} p(s', r \mid s, a) r
\end{aligned}
$$

If the MDP is *ergodic*: the starting state and any early decision made by the agent can have only a temporary effect; in the long run the expectation of being in a state depends only on the policy and the MDP transition probabilities.

Avg. Reward: A Problem Setting for Continuing Tasks

- (Differential) Return:

$$G_t \doteq R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + R_{t+3} - r(\pi) + \cdots.$$

Avg. Reward: A Problem Setting for Continuing Tasks

- (Differential) Return:

$$G_t \doteq R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + R_{t+3} - r(\pi) + \cdots.$$

- Differential value functions:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{r,s'} p(s', r|s, a)\Big[r - r(\pi) + v_\pi(s')\Big],$$

$$q_\pi(s, a) = \sum_{r,s'} p(s', r|s, a)\Big[r - r(\pi) + \sum_{a'} \pi(a'|s') q_\pi(s', a')\Big],$$

$$v_*(s) = \max_a \sum_{r,s'} p(s', r|s, a)\Big[r - \max_\pi r(\pi) + v_*(s')\Big], \text{ and}$$

$$q_*(s, a) = \sum_{r,s'} p(s', r|s, a)\Big[r - \max_\pi r(\pi) + \max_{a'} q_*(s', a')\Big]$$

Avg. Reward: A Problem Setting for Continuing Tasks

- Differential value functions:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{r,s'} p(s', r | s, a)\Big[r - r(\pi) + v_\pi(s')\Big],$$

$$q_\pi(s,a) = \sum_{r,s'} p(s', r | s, a)\Big[r - r(\pi) + \sum_{a'} \pi(a'|s') q_\pi(s', a')\Big],$$

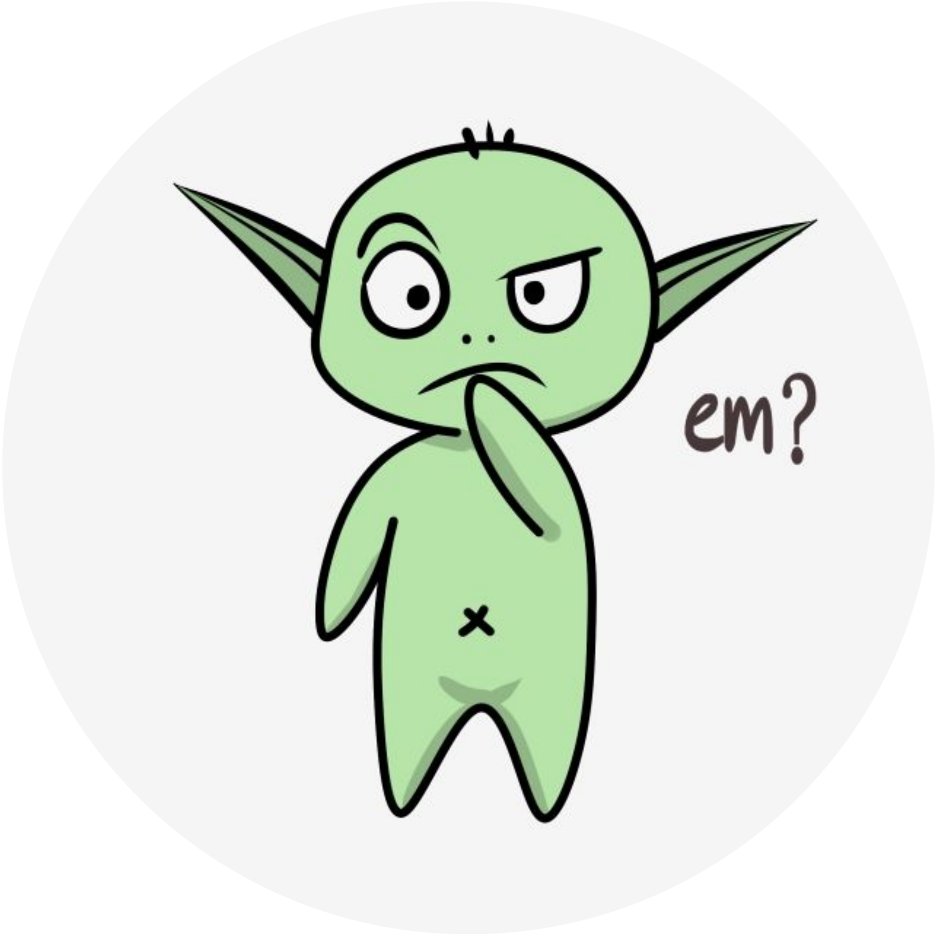
$$v_*(s) = \max_a \sum_{r,s'} p(s', r | s, a)\Big[r - \max_\pi r(\pi) + v_*(s')\Big], \text{ and}$$

$$q_*(s,a) = \sum_{r,s'} p(s', r | s, a)\Big[r - \max_\pi r(\pi) + \max_{a'} q_*(s', a')\Big]$$

- Differential TD error:

$$\delta_t \doteq R_{t+1} - \bar{R}_t + \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t),$$

$$\delta_t \doteq R_{t+1} - \bar{R}_t + \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t).$$



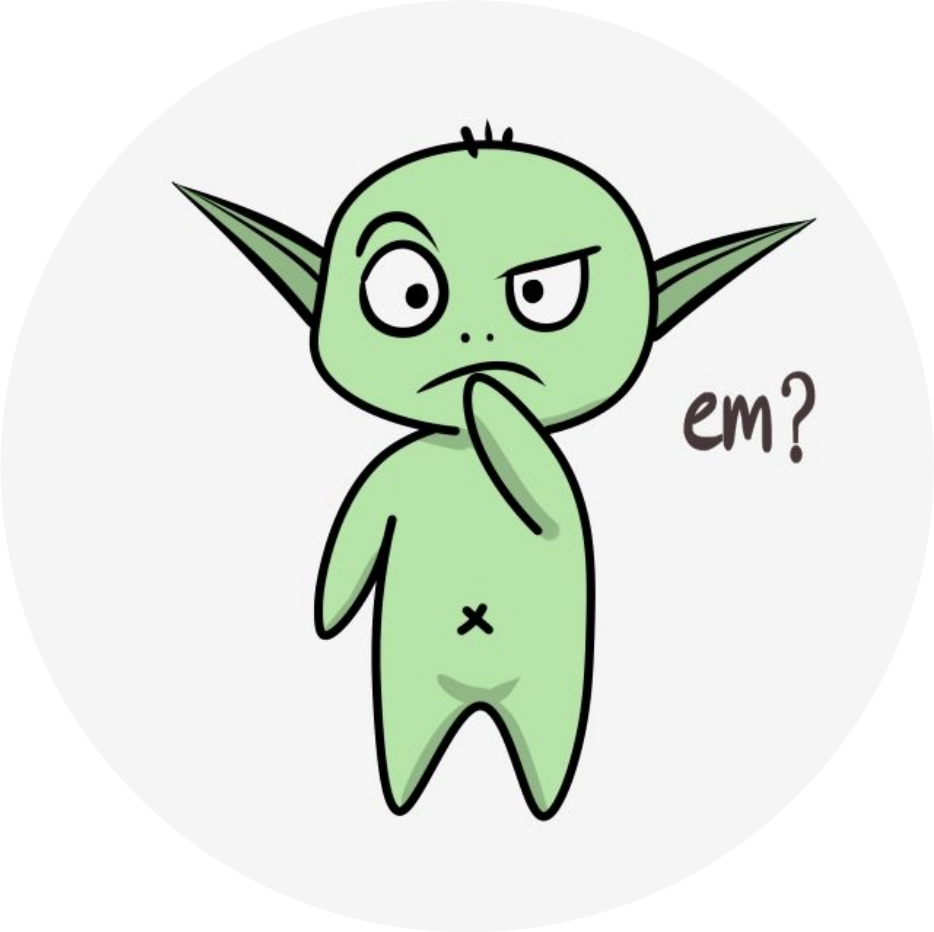
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Differential semi-gradient Sarsa

Differential semi-gradient Sarsa for estimating $\hat{q} \approx q_*$

Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \rightarrow \mathbb{R}$
Algorithm parameters: step sizes $\alpha, \beta > 0$, small $\varepsilon > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
Initialize average reward estimate $\bar{R} \in \mathbb{R}$ arbitrarily (e.g., $\bar{R} = 0$)

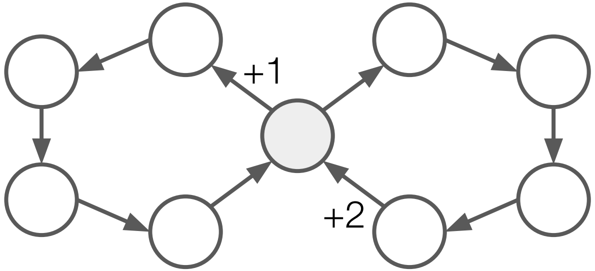
Initialize state S, and action A
Loop for each step:
    Take action A, observe R, S'
    Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)
    $\delta \leftarrow R - \bar{R} + \hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})$
    $\bar{R} \leftarrow \bar{R} + \beta\delta$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S, A, \mathbf{w})$
    $S \leftarrow S'$
    $A \leftarrow A'$



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

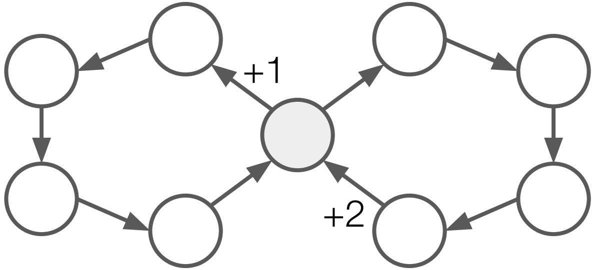
Avg. Reward: A Problem Setting for Continuing Tasks

- Continuing problems without discounting.
  - The agent cares about all rewards equally.



Avg. Reward: A Problem Setting for Continuing Tasks

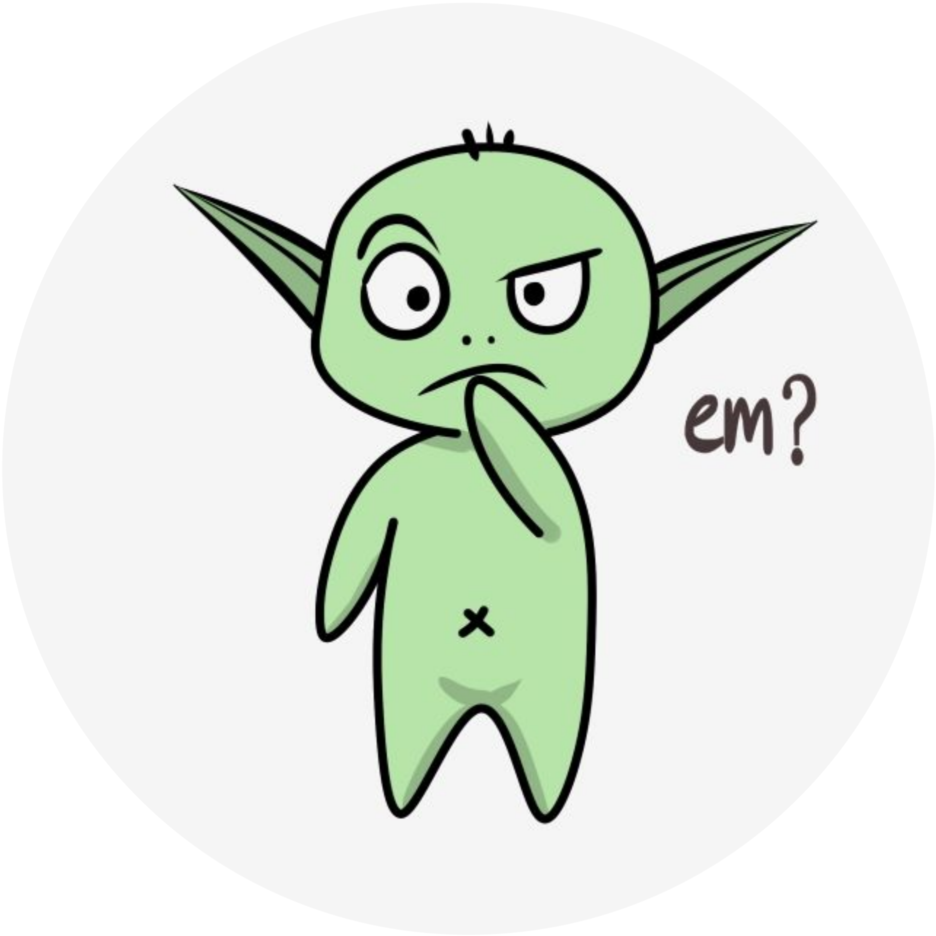
- Continuing problems without discounting.
  - The agent cares about all rewards equally.



- Quality of a policy is defined by the average rate of reward, r(π):

$$
\begin{aligned}
r(\pi) &\doteq \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h} \mathbb{E}[R_t \mid S_0, A_{0:t-1} \sim \pi] \\
&= \lim_{t\to\infty} \mathbb{E}[R_t \mid S_0, A_{0:t-1} \sim \pi], \\
&= \sum_s \mu_\pi(s) \sum_a \pi(a|s) \sum_{s',r} p(s', r \mid s, a) r
\end{aligned}
$$

If the MDP is *ergodic*: the starting state and any early decision made by the agent can have only a temporary effect; in the long run the expectation of being in a state depends only on the policy and the MDP transition probabilities.



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Exercise

In the context of control algorithms *with function approximation*, please provide:

(a) The general form of the update rule for semi-gradient one-step Q-Learning.

(b) The the specific update, with no generic gradient terms, for semi-gradient one-step Q-learning *with linear function approximation*.

Exercise

In the context of control algorithms *with function approximation*, please provide:

(a) The general form of the update rule for semi-gradient one-step Q-Learning.

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \big[R + \gamma \max_{a'} \hat{q}(S', a', \mathbf{w}) - \hat{q}(S, A, \mathbf{w}) \big] \nabla_{\mathbf{w}} \hat{q}(S, A, \mathbf{w})$$

(b) The the specific update, with no generic gradient terms, for semi-gradient one-step Q-learning *with linear function approximation*.

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \big[R + \gamma \max_{a'} \mathbf{w}^\top \mathbf{x}(S', a') - \mathbf{w}^\top \mathbf{x}(S, A) \big] \mathbf{x}(S, A)$$