“A beginning is the time for taking the most delicate care that the balances are correct.”

Frank Herbert, *Dune*

CMPUT 628
Deep RL

Marlos C. Machado

Plan

Overview / Refresher of Model-Free Value-Based Reinforcement Learning

Warning! *This will be quick. It is meant to start establishing a common language between us, but it is too fast for you if you are seeing this for the first time.*

Reminder: You can still leave

- I know, I know, *Deep Reinforcement Learning* sounds fun, modern, and hyp-ey

But…

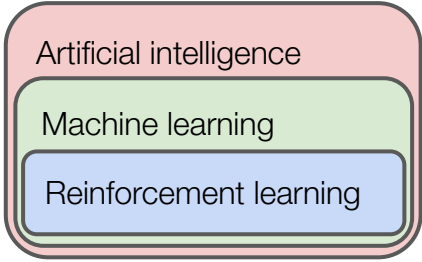
- But this course won't be so well-structured as you (or I) would hope
- I won't teach you how to code fancy deep RL algorithms
- I'm not as much fun as you might think
- I don't care about grades – I might have a reputation :-)
  - *There won't be a practice midterm*
- I don't care if this course ends up being difficult

Please, interrupt me at any time!

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

Reinforcement learning problem formulation

Marlos C. Machado



Reinforcement learning

Reinforcement learning is a computational approach to learning from interaction to maximize a numerical reward signal (Sutton & Barto; 2018)

- The idea of learning by interacting with our environment is very natural
- It is based on the idea of a learning system that wants something, and that adapts its behavior to get that

Some features are unique to reinforcement learning:

- Trial-and-error
- The trade-off between exploration and exploitation
- The delayed credit assignment / delayed reward problem

https://images.squarespace-cdn.com/content/v1/5bdf4a4d7c93279afce735f7/1548721628539-YNH1GBN9MXX3CQMCSYVR/treat+dog+in+background.jpg?format

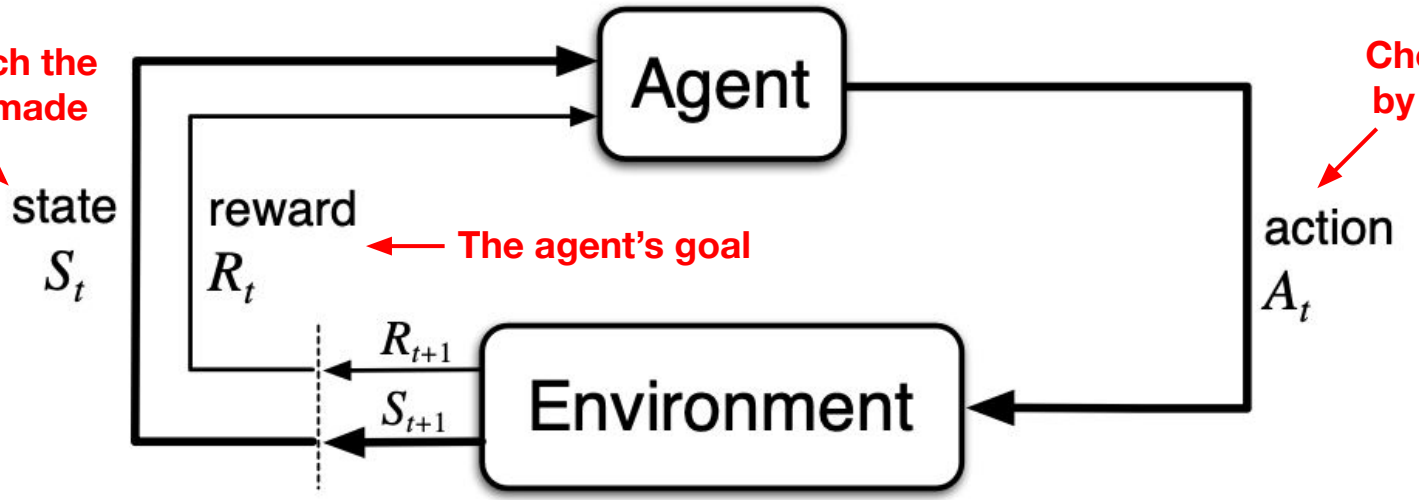
Reinforcement learning (RL)

- RL is about learning from *evaluative* feedback (an evaluation of the taken actions) rather than *instructive* feedback (being given the correct actions).
  - Exploration is essential in reinforcement learning.

- It is not necessarily about online learning, as it is sometimes said, but more generally about sequential decision-making.

- Reinforcement learning potentially allows for continual learning but in practice, quite often we deploy our systems.
  - Continual learning is important, but this course is not about this.

The Agent-Environment Interface



Basis on which the choices are made

Choice made by the agent

The agent's goal

Figure 3.1: The agent–enviro

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$$

The ultimate goal: Maximize Returns

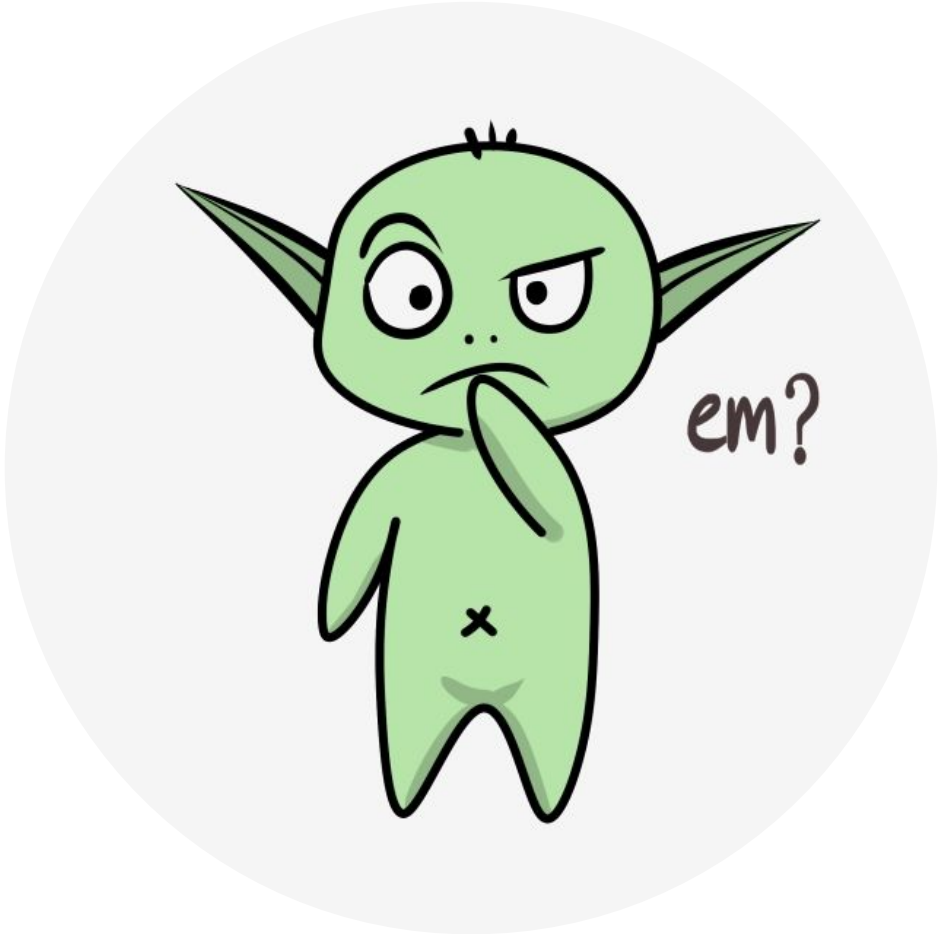
End of an episode

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

Continuing task

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots \right) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$



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

Tabular value-based model-free reinforcement learning

Marlos C. Machado

Value Functions and Policies

- *Value functions are "functions of states (or state-action pairs) that estimate how good it is for the agent to be in a given state".*
- "How good" means expected return.
- Expected returns depend on how the agent behaves, that is, its *policy*.

Policy

- A policy is a mapping from states to probabilities of selecting each possible action:

$$\pi : \mathcal{S} \to \Delta(\mathcal{A})$$

in other words, π(a|s) is the probability that $A_t = a$ if $S_t = s$.

Value Function

- The value function of a state s under a policy π, denoted $v_\pi(s)$ or $q_\pi(s, a)$, is the expected return when starting in s, taking a (for q_π), and following π thereafter.

state-value function for policy π

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right]$$

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right]$$

action-value function for policy π

Optimal Policies and Optimal Value Functions

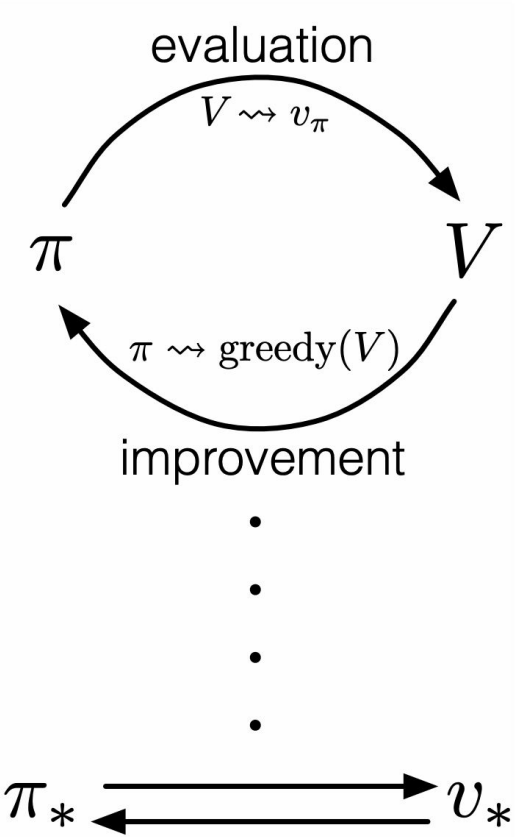
- Value functions define a partial ordering over policies.
  - $\pi \geq \pi'$ iff $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$.
  - There is always at least one policy that is better than or equal to all other policies. The *optimal policy*.

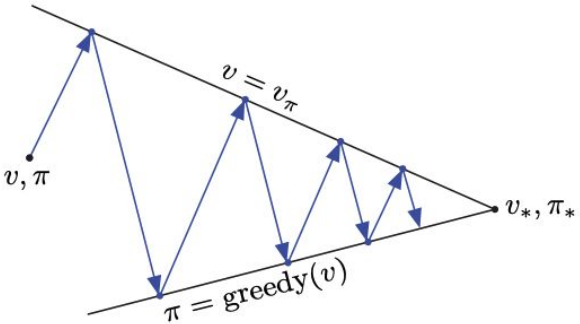
$$v_*(s) \doteq \max_\pi v_\pi(s)$$

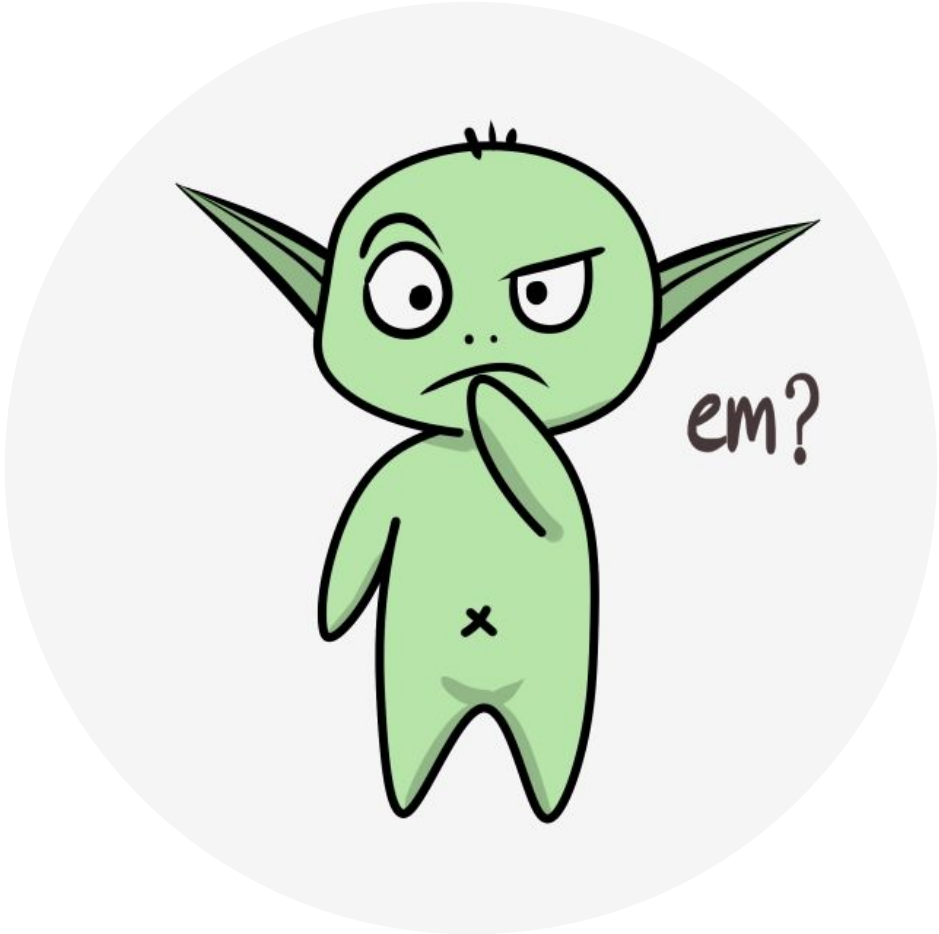
$$q_*(s,a) \doteq \max_\pi q_\pi(s,a)$$

$$q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$$

Generalized Policy Iteration







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

TD Prediction

A simple every-visit Monte Carlo method is:

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[G_t - V(S_t)\Big]$$

What if we don't want to wait until we have a full return (end of episode)!

$$NewEstimate \leftarrow OldEstimate \; + \; StepSize \; \Big[Target - OldEstimate\Big]$$

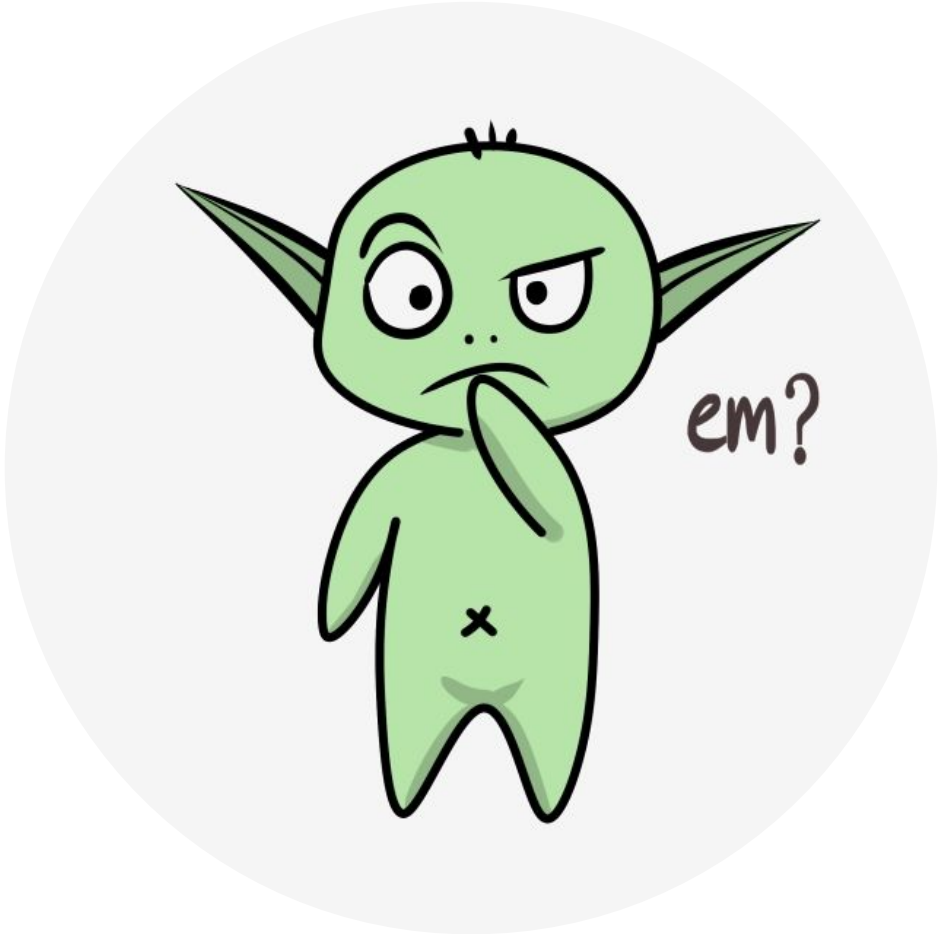
TD Prediction

A simple every-visit Monte Carlo method is:

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[\underbrace{G_t}_{\textbf{Target}} - V(S_t)\Big]$$

Temporal-Difference Learning:

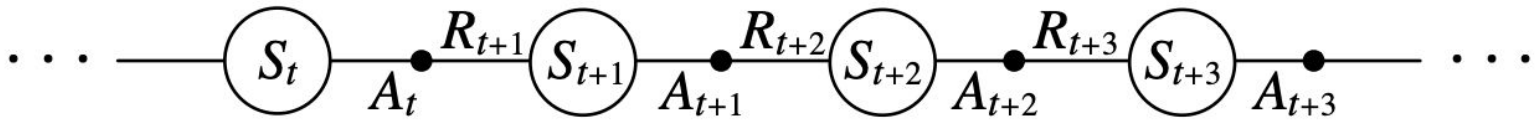
$$V(S_t) \leftarrow V(S_t) + \alpha\Big[\underbrace{R_{t+1} + \gamma V(S_{t+1})}_{\textbf{Target}} - V(S_t)\Big]$$



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

Sarsa: On-policy Control

- We again use generalized policy iteration (GPI), but now using TD for evaluation.
- We need to learn an action-value function instead of a state-value function.



$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \Big]$$

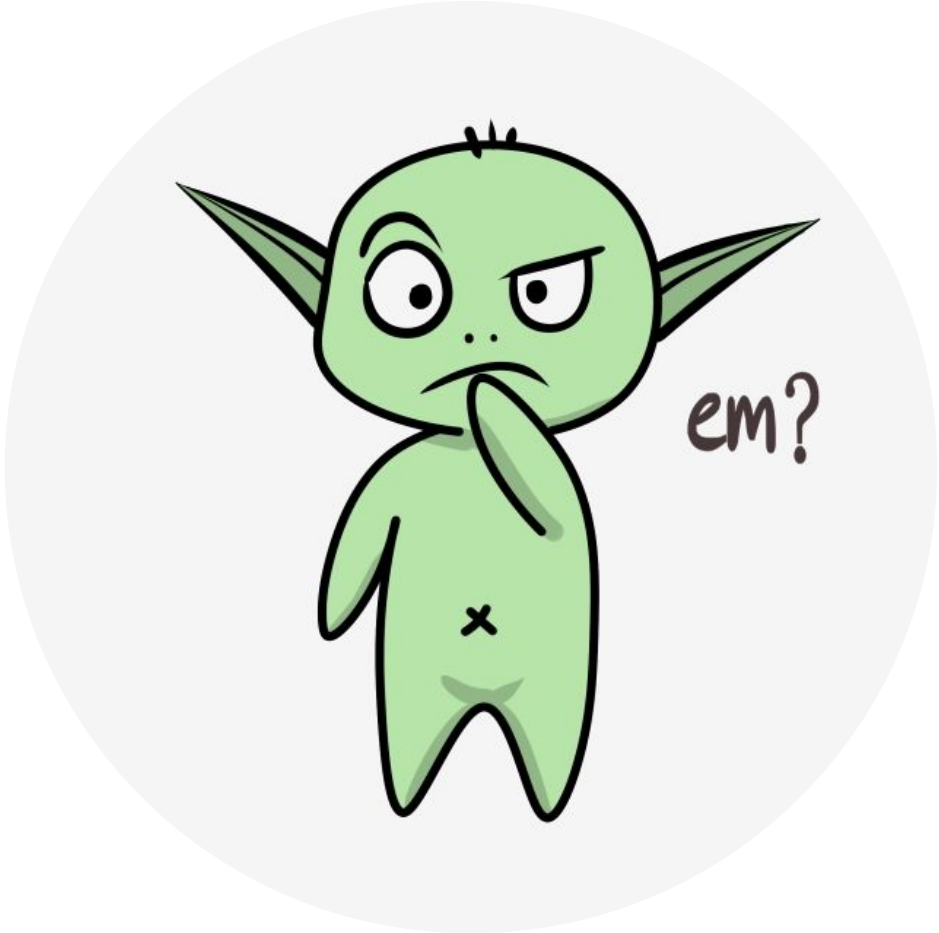
Sarsa: On-policy Control

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

We need to explore!



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

Q-Learning: Off-Policy Control

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \Big]$$

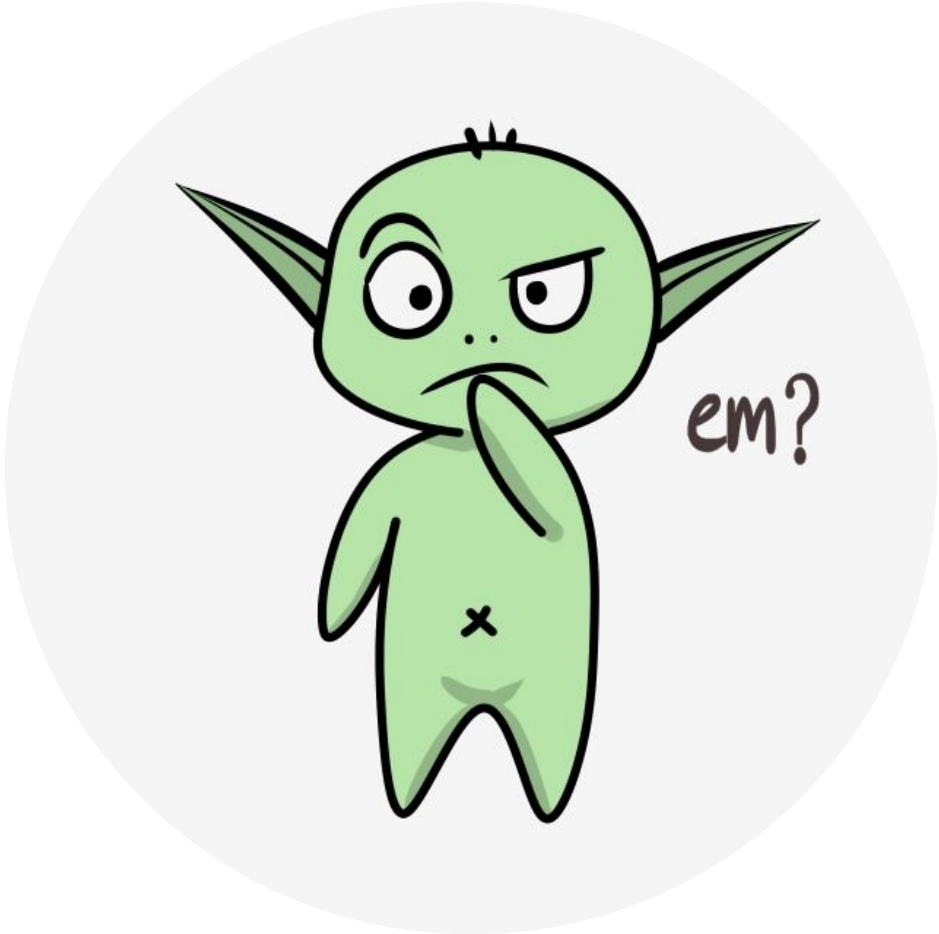
- Q directly approximates q_*, regardless of the policy being followed.
- Notice we do not need importance sampling. We are updating a state–action pair. We do not have to care how likely we were to select the action; now that we have selected it we want to learn fully from what happens.

Q-Learning: Off-Policy Control

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

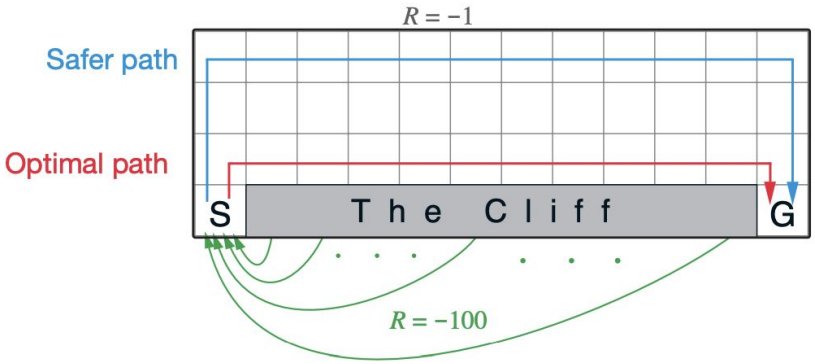
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

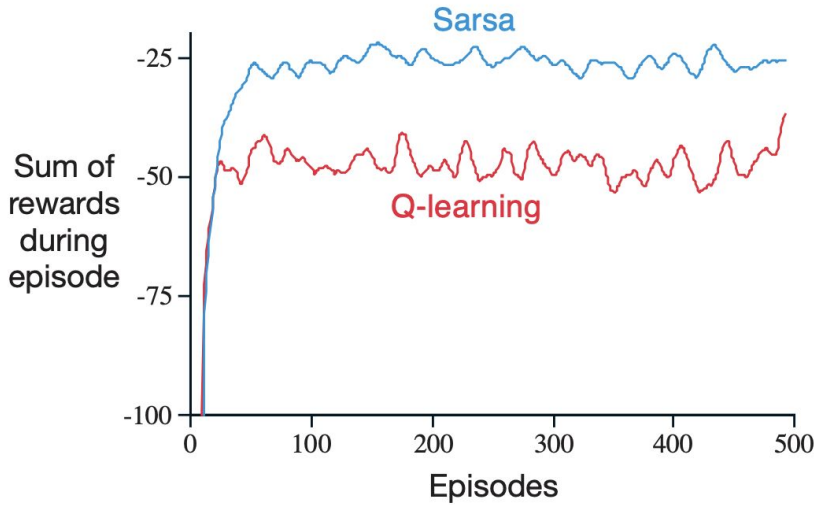
Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

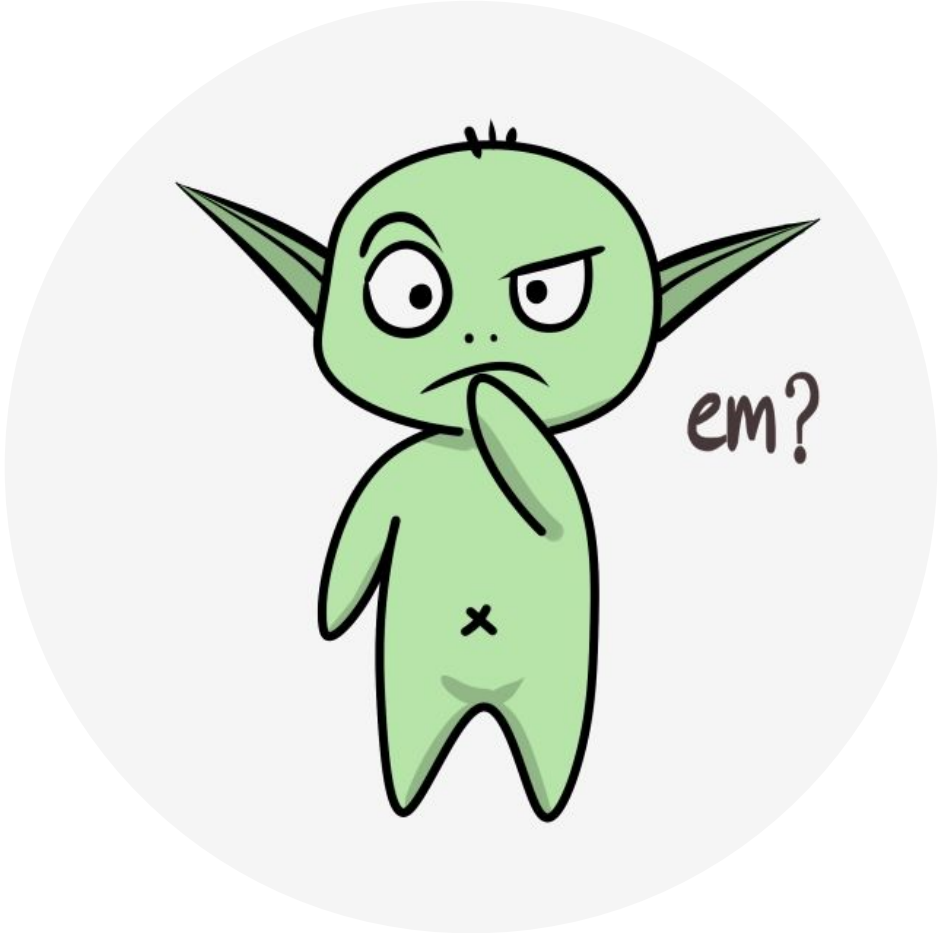


https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Example – Q-Learning vs Sarsa







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

Model-free value-based reinforcement learning with function approximation

Marlos C. Machado

The Prediction Objective (A Notion of Accuracy)

- In the tabular case we can have equality, but with FA, not anymore.
  - Making one state's estimate more accurate invariably means making others' less accurate.
- Mean Squared Error:

$$\overline{\text{VE}}(\mathbf{w}) \doteq \sum_{s \in \mathcal{S}} \mu(s) \Big[v_\pi(s) - \hat{v}(s,\mathbf{w}) \Big]^2$$

How much do we care about the error in each state s.

Usually, the fraction of time spent in s.
On-policy distribution.

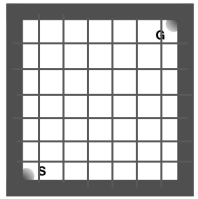
Recipe for Deriving a Concrete Algorithm for SGD

1. Specify a function approximation architecture (parametric form of v_π).
2. Write down your objective function.
3. Take the derivative of the objective function with respect to the weights.
4. Simplify the general gradient expression for your parametric form.
5. Make a weight update rule:

   $w = w - \alpha$ `GRAD`

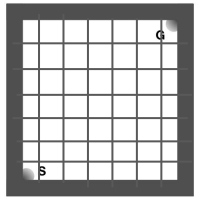
1. Specify a FA architecture (parametric form of v_π)

- We will use *state aggregation with linear function approximation*



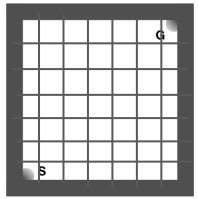
1. Specify a FA architecture (parametric form of v_π)

- We will use *state aggregation with linear function approximation*
- State aggregation
  - The features are always binary with only a single active feature that is not zero



1. Specify a FA architecture (parametric form of v_π)

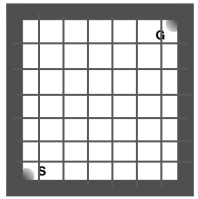
- We will use *state aggregation with linear function approximation*
- State aggregation
  - The features are always binary with only a single active feature that is not zero
- Value function
  - Linear function



$$v_\pi(s) \approx \hat{v}(s, \mathbf{w}) \doteq \mathbf{w}^\top \mathbf{x}(s) \doteq \sum_{i=1}^{d} w_i \cdot x_i(s)$$

2. Write down your objective function

- We will use the *value error*



$$\begin{aligned}\overline{\text{VE}}(\mathbf{w}) &\doteq \sum_{s\in\mathcal{S}} \mu(s)\Big[v_\pi(s) - \hat{v}(s,\mathbf{w})\Big]^2 \\ &= \sum_{s\in\mathcal{S}} \mu(s)\Big[v_\pi(s) - \mathbf{w}^\top\mathbf{x}(s)\Big]^2\end{aligned}$$

3. Take the derivative of the obj. function w.r.t. the weights

$$
\begin{aligned}
\nabla\overline{\text{VE}}(\mathbf{w}) &= \nabla \sum_{s\in\mathcal{S}} \mu(s)\Big[v_\pi(s) - \mathbf{w}^\top\mathbf{x}(s)\Big]^2 \\
&= \sum_{s\in\mathcal{S}} \mu(s)\nabla\Big[v_\pi(s) - \mathbf{w}^\top\mathbf{x}(s)\Big]^2 \\
&= -\sum_{s\in\mathcal{S}} \mu(s)2\Big[v_\pi(s) - \mathbf{w}^\top\mathbf{x}(s)\Big]\nabla\mathbf{w}^\top\mathbf{x}(s)
\end{aligned}
$$

4. Simplify the general gradient expression

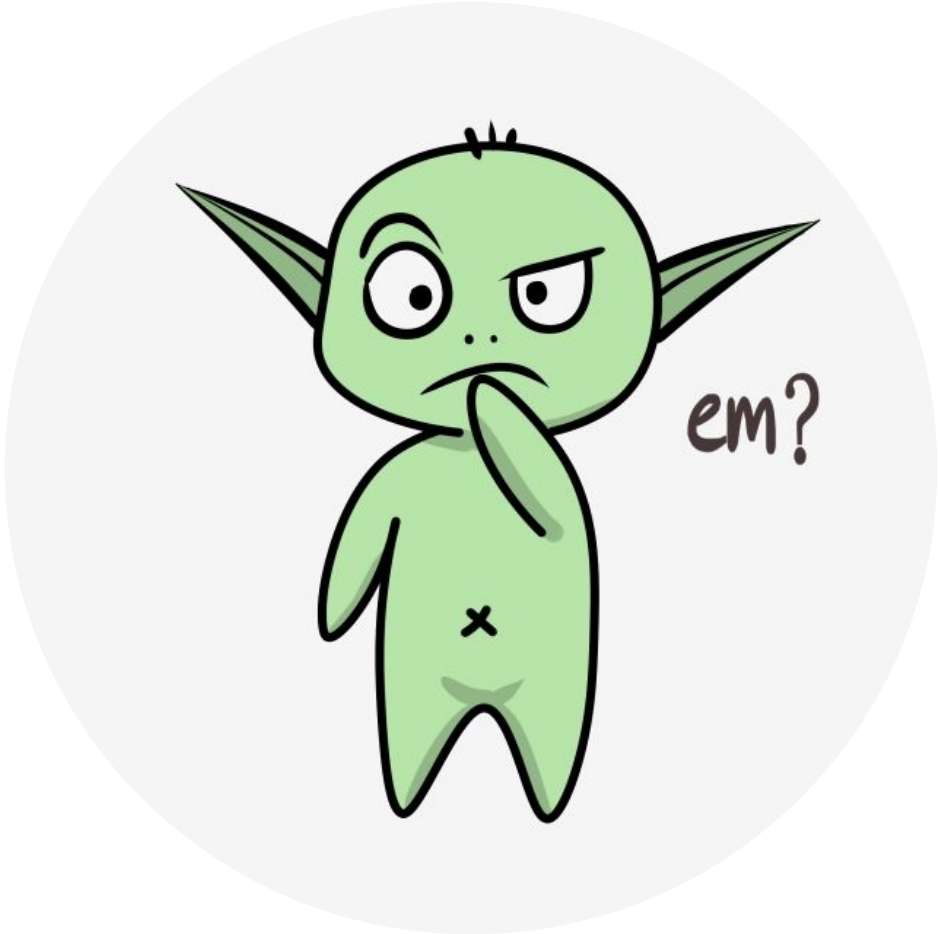
$$\begin{aligned}\nabla\overline{\text{VE}}(\mathbf{w}) &= -\sum_{s\in\mathcal{S}}\mu(s)2\Big[v_\pi(s)-\mathbf{w}^\top\mathbf{x}(s)\Big]\nabla\mathbf{w}^\top\mathbf{x}(s)\\ &= -\sum_{s\in\mathcal{S}}\mu(s)2\Big[v_\pi(s)-\mathbf{w}^\top\mathbf{x}(s)\Big]\mathbf{x}(s)\end{aligned}$$

$$\nabla\mathbf{w}^\top\mathbf{x}(s)=\mathbf{x}(s)$$

5. Make a weight update rule

$$\nabla \overline{\text{VE}}(\mathbf{w}) = -\sum_{s \in \mathcal{S}} \mu(s) 2 \Big[v_\pi(s) - \mathbf{w}^\top \mathbf{x}(s) \Big] \mathbf{x}(s)$$

$$\begin{aligned} \mathbf{w}_{t+1} &= \mathbf{w}_t + \alpha 2 \Big[v_\pi(s) - \mathbf{w}^\top \mathbf{x}(s) \Big] \mathbf{x}(s) \\ &= \mathbf{w}_t + \alpha \Big[v_\pi(s) - \mathbf{w}^\top \mathbf{x}(s) \Big] \mathbf{x}(s) \end{aligned}$$



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

A More Realistic Update

- Let U_t denote the *t*-th training example, $S_t \mapsto v_\pi(S_t)$, of some (possibly random), approximation to the true value.

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

Gradient Monte Carlo Algorithm for Estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameter: step size $\alpha > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, S_1, A_1, \ldots, R_T, S_T$ using π
  Loop for each step of episode, $t = 0, 1, \ldots, T-1$:
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha \big[G_t - \hat{v}(S_t, \mathbf{w}) \big] \nabla \hat{v}(S_t, \mathbf{w})$

Semi-gradient TD

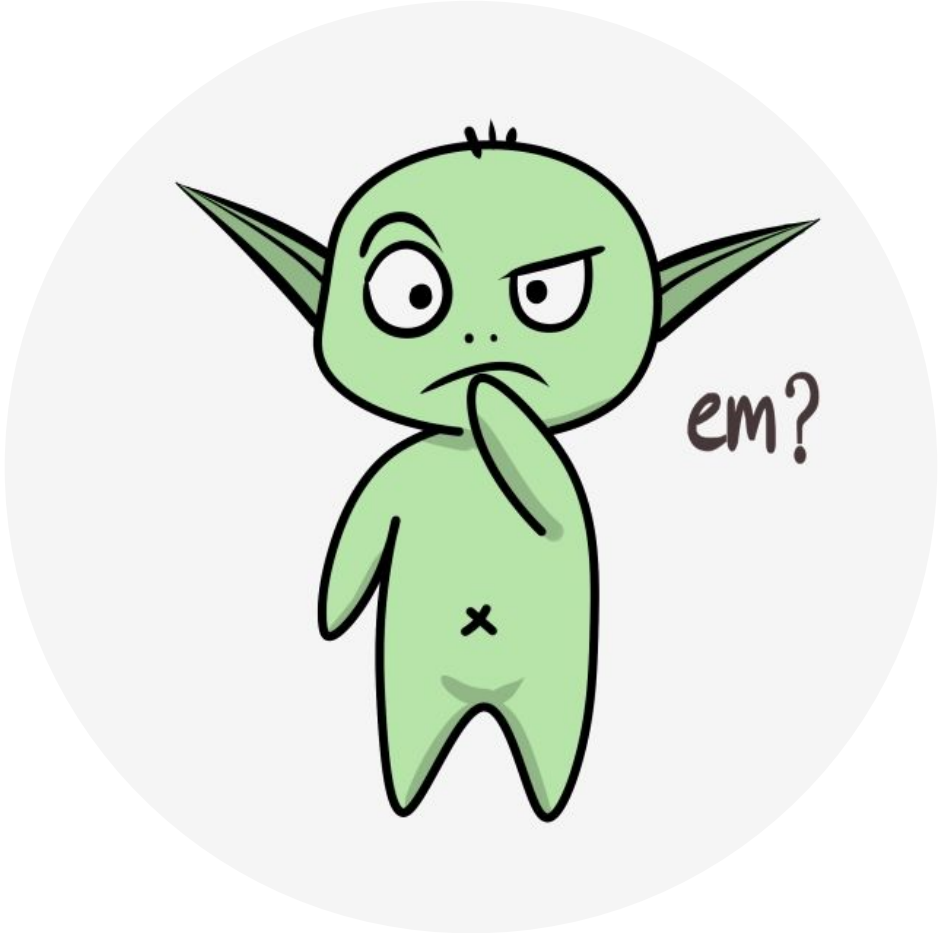
- What if $U_t \doteq R_{t+1} + \gamma\, \hat{v}(S_{t+1}, \mathbf{w}_t)$?
- We lose several guarantees when we use a bootstrapping estimate as target.
  - The target now also depends on the value of $\mathbf{w}_t$, so the target is not independent of $\mathbf{w}_t$.
- Bootstrapping are not instances of true gradient descent. They take into account the effect of changing the weight vector $\mathbf{w}_t$ on the estimate, but ignore its effect on the target. Thus, they are a *semi-gradient method*.
- Regardless of the theoretical guarantees, we use them all the time ¯_(ツ)_/¯

Semi-gradient TD

Semi-gradient TD(0) for estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^d \to \mathbb{R}$ such that $\hat{v}(\text{terminal},\cdot) = 0$
Algorithm parameter: step size $\alpha > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        Choose $A \sim \pi(\cdot|S)$
        Take action A, observe R, S'
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})\big]\nabla\hat{v}(S,\mathbf{w})$
        $S \leftarrow S'$
    until S is terminal



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, $A_t \mapsto U_t$.
- 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*:

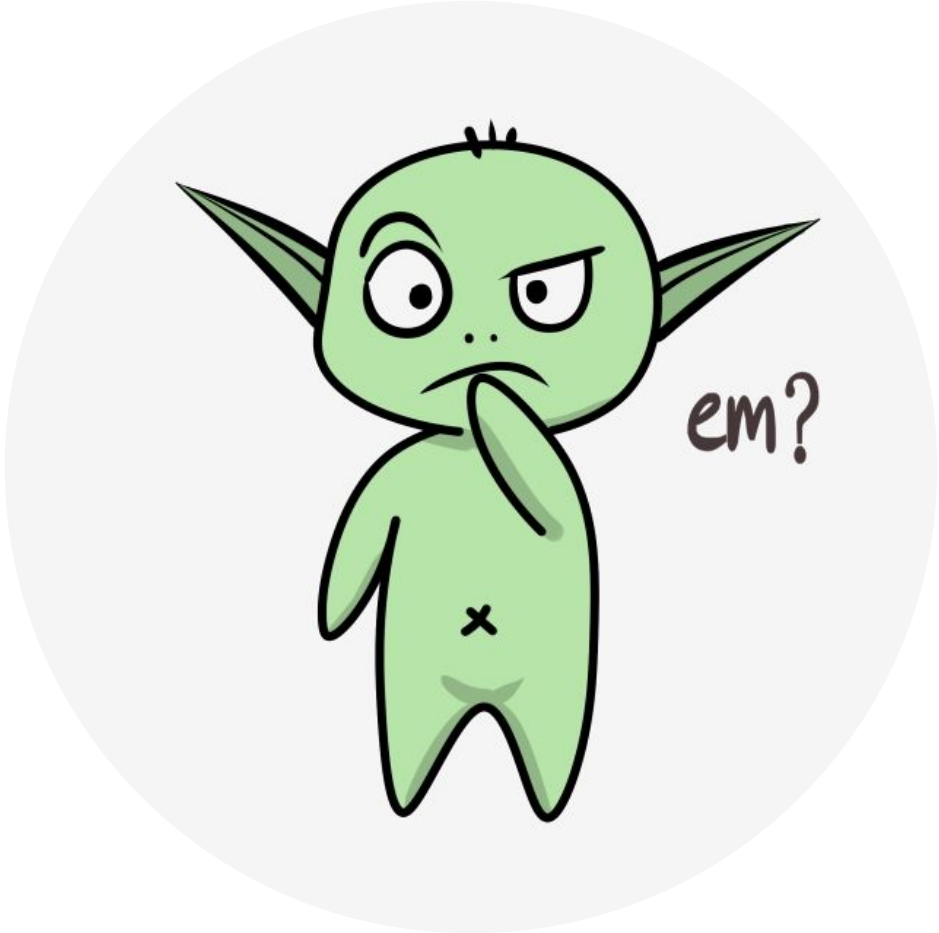
$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \Big[R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t) \Big] \nabla \hat{q}(S_t, A_t, \mathbf{w}_t)$$

Episodic Semi-gradient Sarsa

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

Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
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)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
    $S \leftarrow S'$
    $A \leftarrow A'$



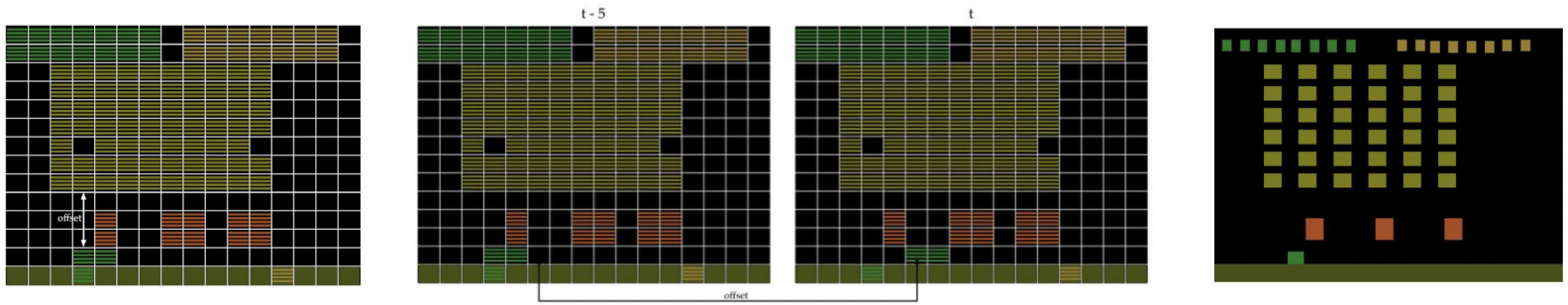
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

This works!

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

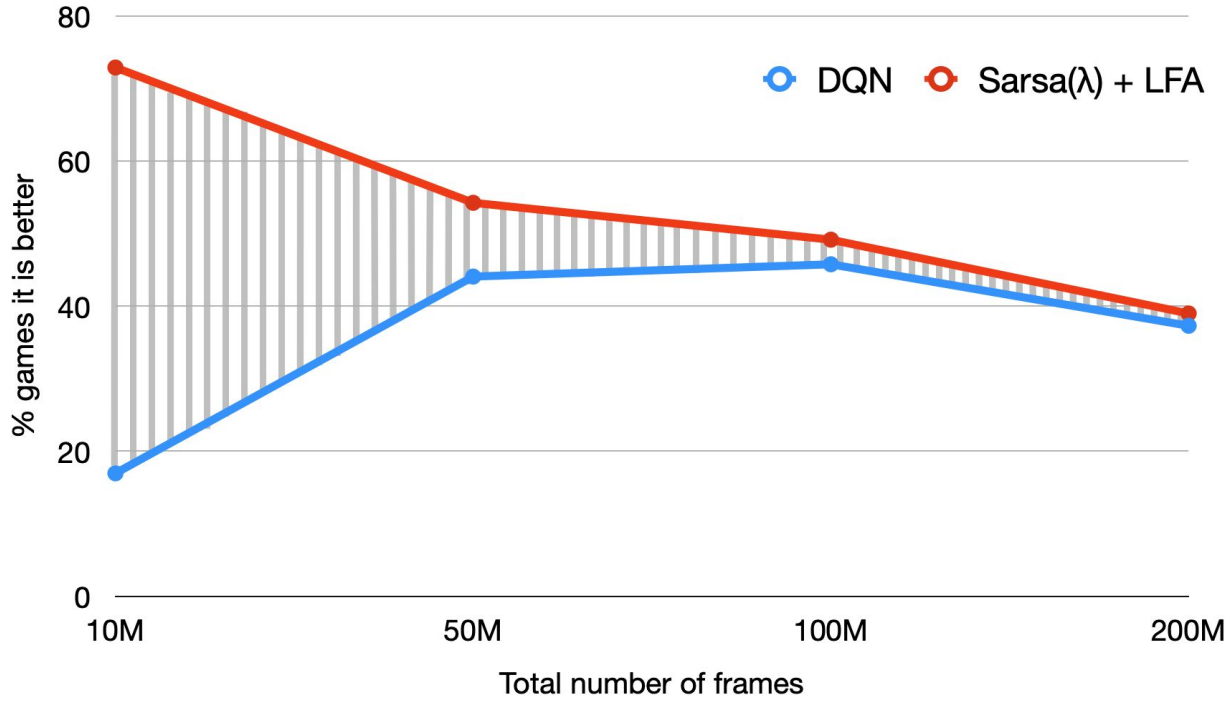
Yitao Liang[†], Marlos C. Machado[‡], Erik Talvitie[†], and Michael Bowling[‡]
[†]Franklin & Marshall College
Lancaster, PA, USA
{yliang, erik.talvitie}@fandm.edu

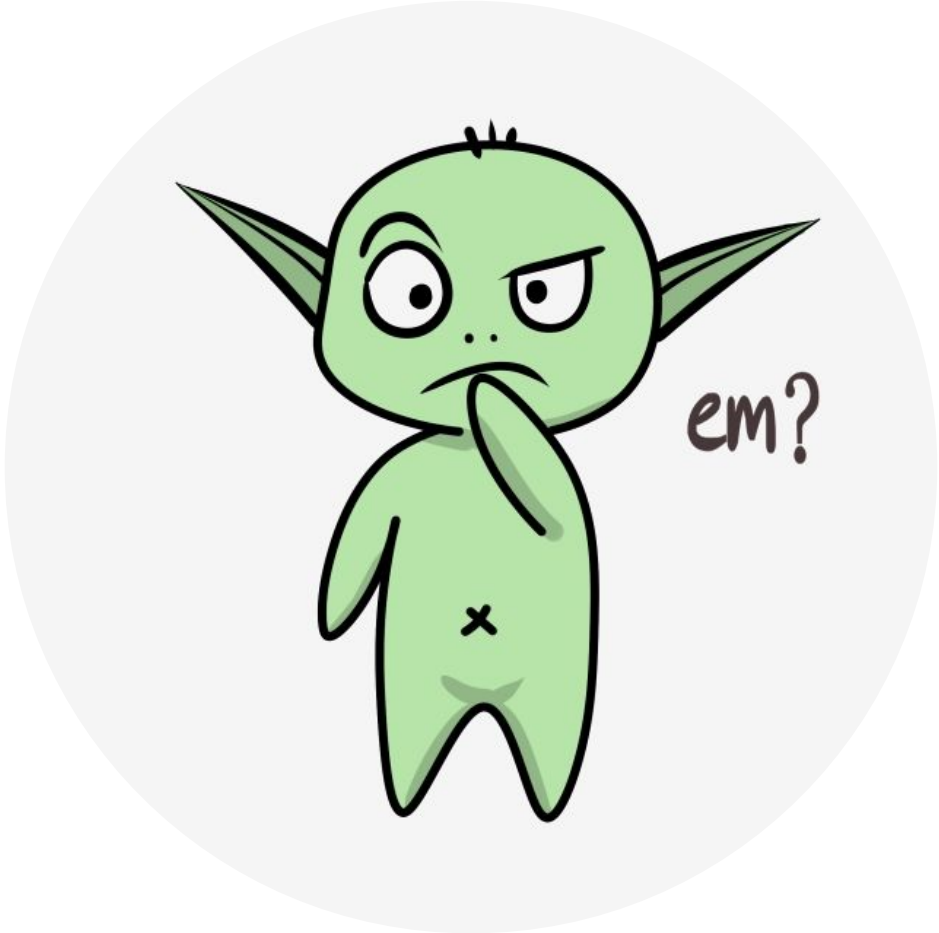
[‡]University of Alberta
Edmonton, AB, Canada
{machado, mbowling}@ualberta.ca



There are many trade-offs, we need to understand them

[Liang et al., 2016; Machado et al. 2018]





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

Next class

- What I plan to do:
  - Start talking about Deep RL; more specifically, DQN.

- What I recommend YOU to do for next class:
  - Read the DQN paper: V. Mnih et al.: Human-level control through deep reinforcement learning. Nature 518(7540): 529-533 (2015).
  - Read evaluation paper: M. C. Machado et al.: Revisiting the Arcade Learning Environment: Evaluation protocols and open problems for general agents. J. Artif. Intell. Res. 61: 523-562 (2018).