
“For even the very wise cannot see all ends.”
J.R.R. Tolkien, The Fellowship of the Ring
CMPUT 365
Introduction to RL
Marlos C. Machado
Classes 22-24/35

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 are due on Wednesday.
- The grades for Module 2 of Coursera are now available on Canvas. If you have any questions / concerns, please contact cmput365@ualberta.ca.
- We will be marking Midterm 2 this next week.
  - Exam viewing instructions will be posted after grades are released.
- CNAS has confirmed the date and location of our final exam.
  https://www.ualberta.ca/en/registrar/examinations/exam-schedules/fall-winter-exam-schedule.html?search=CMPUT%20365
  - Thursday, April 16 at 13:00 at CCIS 1-430.
  - *It will be 2-hours long*.
  - It will cover the whole course.

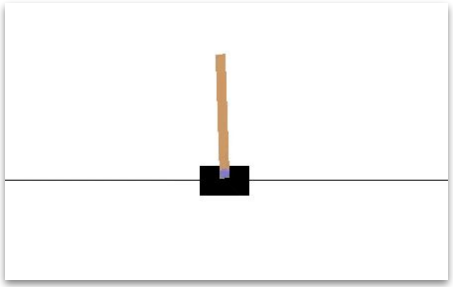
Please, interrupt me at any time!

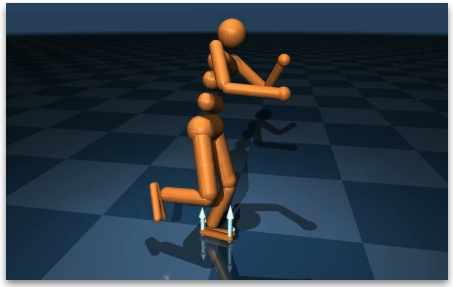
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Part II: Approximate Solution Methods

What We Have Done So Far: *Tabular RL*

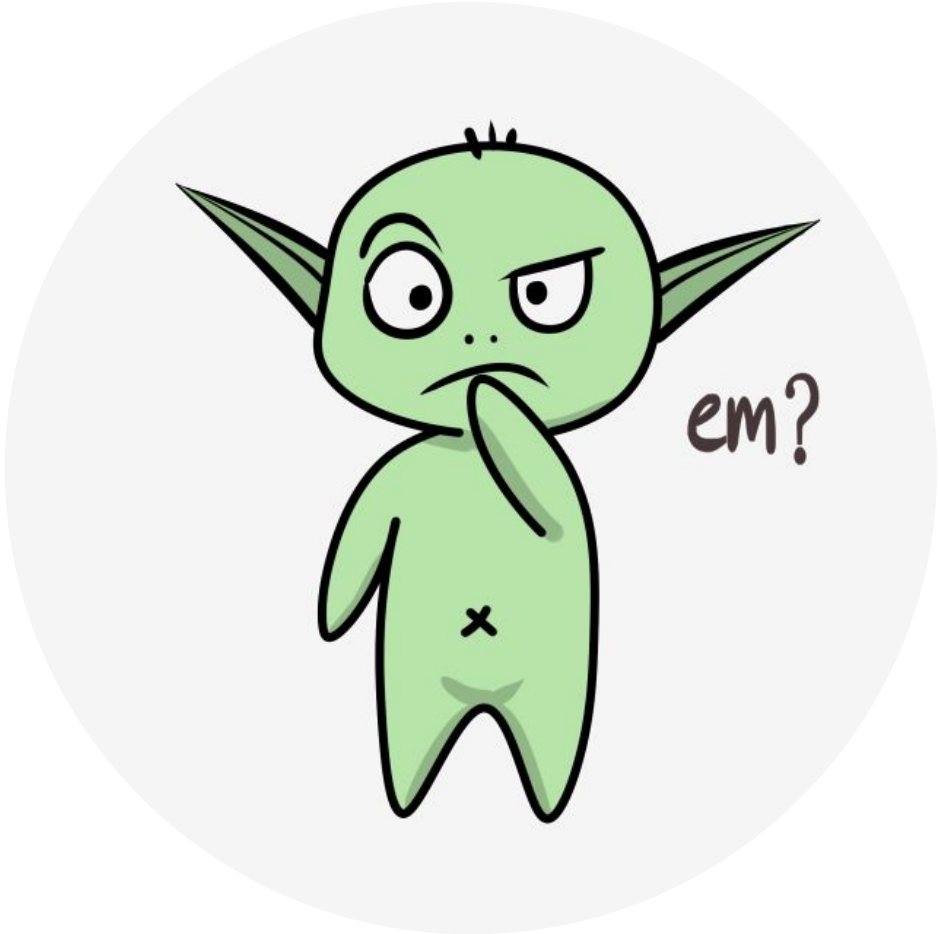
- Absolutely everything we did was in the *tabular* case.
  - There's a huge memory cost in having to fill a table with a ridiculously large number of states.
  - We might never see the same state twice.
  - We cannot expect to find an optimal policy (or value function) even in the limit of infinite time and data.
- What about…



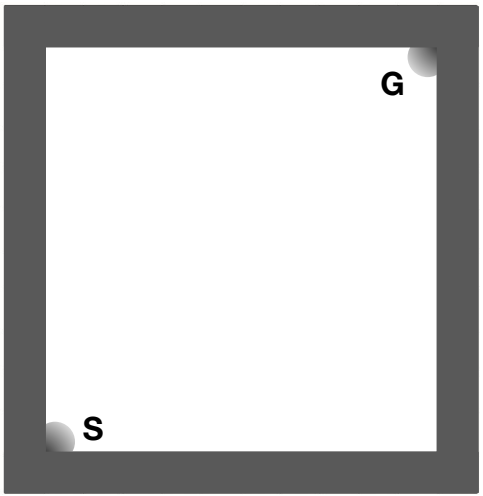
From Now On: *Generalization*

- Instead, we should find a good *approximate* solution using limited computational resources.
- We need to generalize from previous encounters with different states that are in some sense similar to the current one.
- We obtain generalization with *function approximation* (often from the supervised learning literature).



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

Function Approximation – An Example



State space: <x, y> coordinates (continuous, no grid) and <$\dot{x}$, $\dot{y}$> velocity (continuous).

Start state: Somewhere in the bottom left corner, where a suitable <x, y> coordinate is selected randomly.

Action space: Adding or subtracting a small force to $\dot{x}$ velocity or $\dot{y}$ velocity, or leaving them unchanged.

Dynamics: Traditional physics + noise, collisions with obstacles are fully elastic and cause the agent to bounce.

Reward function: +1 when you hit the region in G.

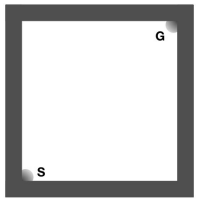
γ: 0.9.

Function Approximation

- In the tabular case, $\mathbf{v}_\pi \in \mathbb{R}^{|\mathcal{S}|}$.
- Instead, we will approximate $\mathbf{v}_\pi$ using a function parameterized by some weights $\mathbf{w} \in \mathbb{R}^d$ where $d \ll |\mathcal{S}|$. We will write $\hat{\mathbf{v}}(s,\mathbf{w}) \approx v_\pi(s)$.

Function Approximation

- In the tabular case, $\mathbf{v}_\pi \in \mathbb{R}^{|\mathcal{S}|}$.
- Instead, we will approximate $\mathbf{v}_\pi$ using a function parameterized by some weights $\mathbf{w} \in \mathbb{R}^d$ where $d \ll |\mathcal{S}|$. We will write $\hat{\mathbf{v}}(s,\mathbf{w}) \approx v_\pi(s)$.
- An example:



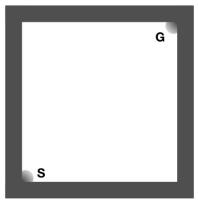
$$\mathbf{s} = \begin{pmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{pmatrix} \qquad \mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} \qquad \hat{v}(\mathbf{s}, \mathbf{w}) = \mathbf{s}^\top \mathbf{w}$$

Generalization: When something changes, many states can be affected.

Feature vector

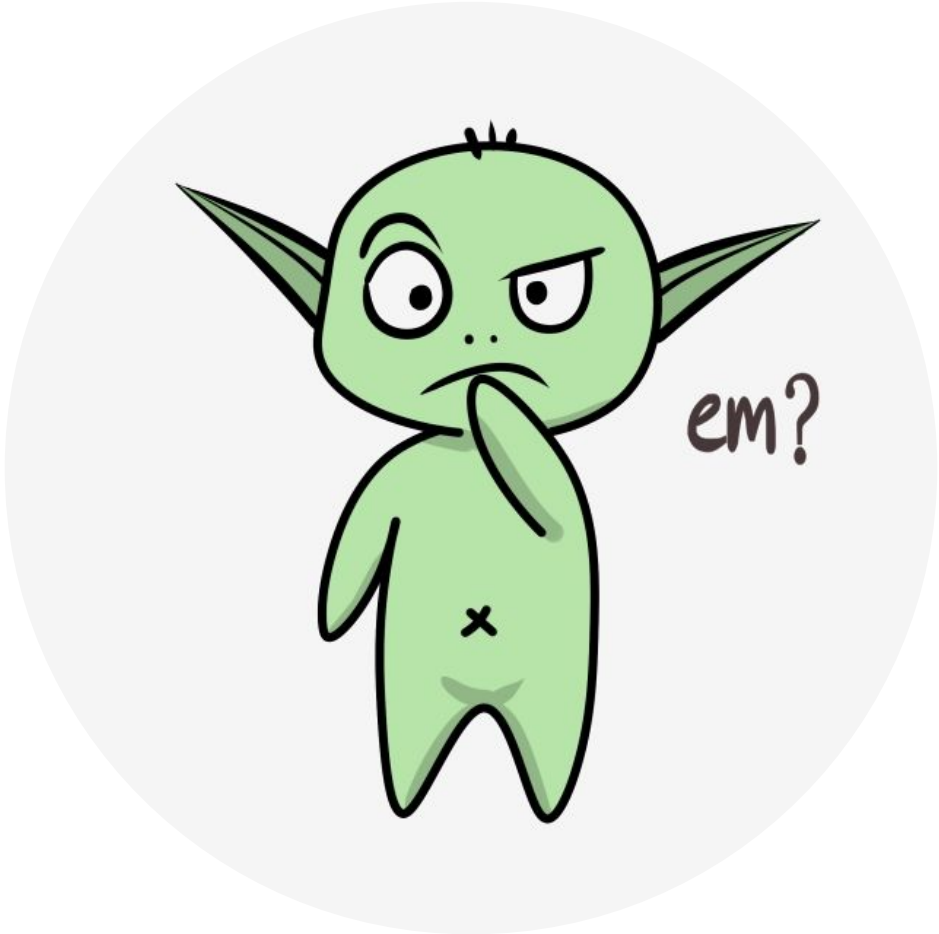
Function Approximation

- In the tabular case, $\mathbf{v}_\pi \in \mathbb{R}^{|\mathcal{S}|}$.
- Instead, we will approximate $\mathbf{v}_\pi$ using a function parameterized by some weights $\mathbf{w} \in \mathbb{R}^d$ where $d \ll |\mathcal{S}|$. We will write $\hat{\mathbf{v}}(s,\mathbf{w}) \approx v_\pi(s)$.
- An example:



$$\mathbf{s} = \begin{pmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{pmatrix} \qquad \mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} \qquad \hat{v}(\mathbf{s}, \mathbf{w}) = \mathbf{s}^\top \mathbf{w}$$

- Extending RL to function approximation also makes it applicable to partially observable problems, in which the full state is not available to the agent.
  I may use **o** to denote the agent's observation (instead of **s**).



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

Chapter 9

On-policy Prediction with Approximation

Prediction

Value-function Approximation

- We can interpret each update we have seen so far as an example of the desired input-output behavior of the value function.
- Let $s \mapsto u$ denote an individual update, where s is the state updated and u the update target that s's estimated value is shifted to.
  - $S_t \mapsto G_t$ **Monte Carlo update**
  - $S_t \mapsto R_{t+1} + \gamma\, v_\pi(S_{t+1})$ **TD(0)**
  - $s \mapsto \mathbb{E}_\pi\,[R_{t+1} + \gamma\, v_\pi(S_{t+1}) \mid S_t = s]$ **Dynamic Programming**
- Supervised learning methods learn to mimic input-output examples, and when the output are numbers, the process is called *function approximation*.

A note from the textbook

Viewing each update as a conventional training example in this way enables us to use any of a wide range of existing function approximation methods for value prediction. In principle, we can use any method for supervised learning from examples, including artificial neural networks, decision trees, and various kinds of multivariate regression. However, not all function approximation methods are equally well suited for use in reinforcement learning. The most sophisticated artificial neural network and statistical methods all assume a static training set over which multiple passes are made. In reinforcement learning, however, it is important that learning be able to occur online, while the agent interacts with its environment or with a model of its environment. To do this requires methods that are able to learn efficiently from incrementally acquired data. In addition, reinforcement learning generally requires function approximation methods able to handle nonstationary target functions (target functions that change over time). For example, in control methods based on GPI (generalized policy iteration) we often seek to learn q_π while π changes. Even if the policy remains the same, the target values of training examples are nonstationary if they are generated by bootstrapping methods (DP and TD learning). Methods that cannot easily handle such nonstationarity are less suitable for reinforcement learning.

Not really!

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.

The Prediction Objective (A Notion of Accuracy)

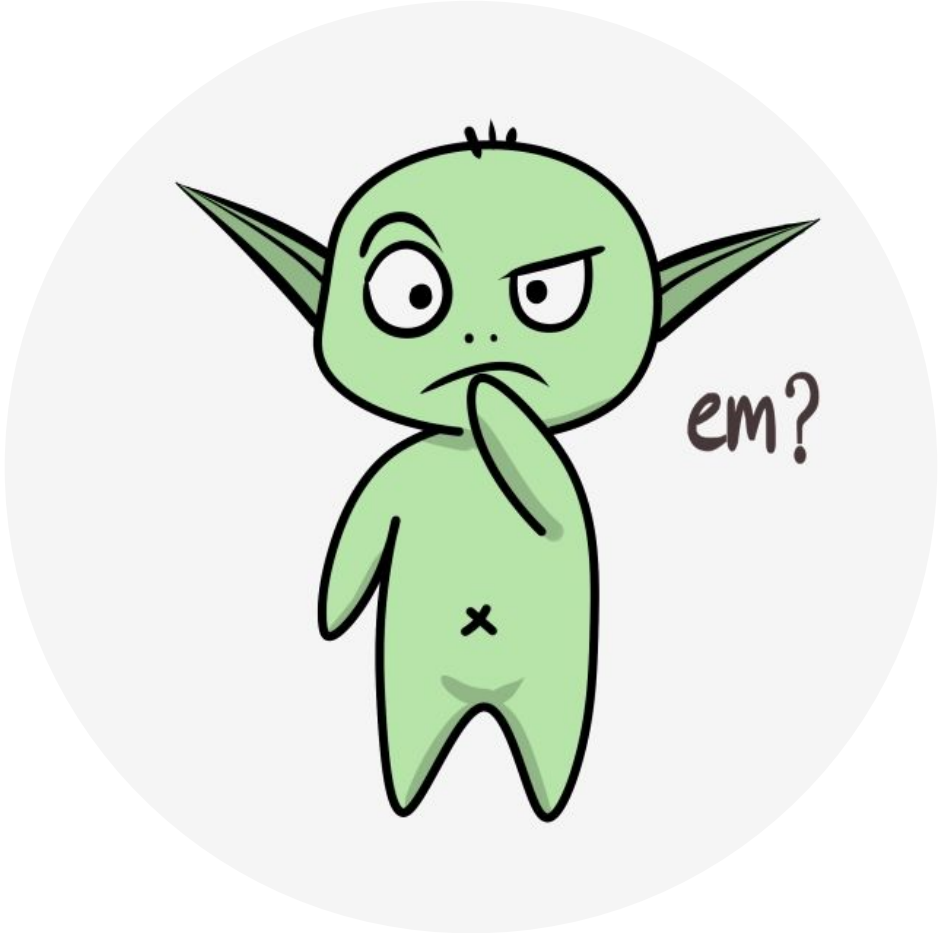
- In the tabular case we can have equality, but with FA, not anymore.
  - Making one state's estimate more accurate invariably means making others' less accurate.
- Mean Squared Error:

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

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

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

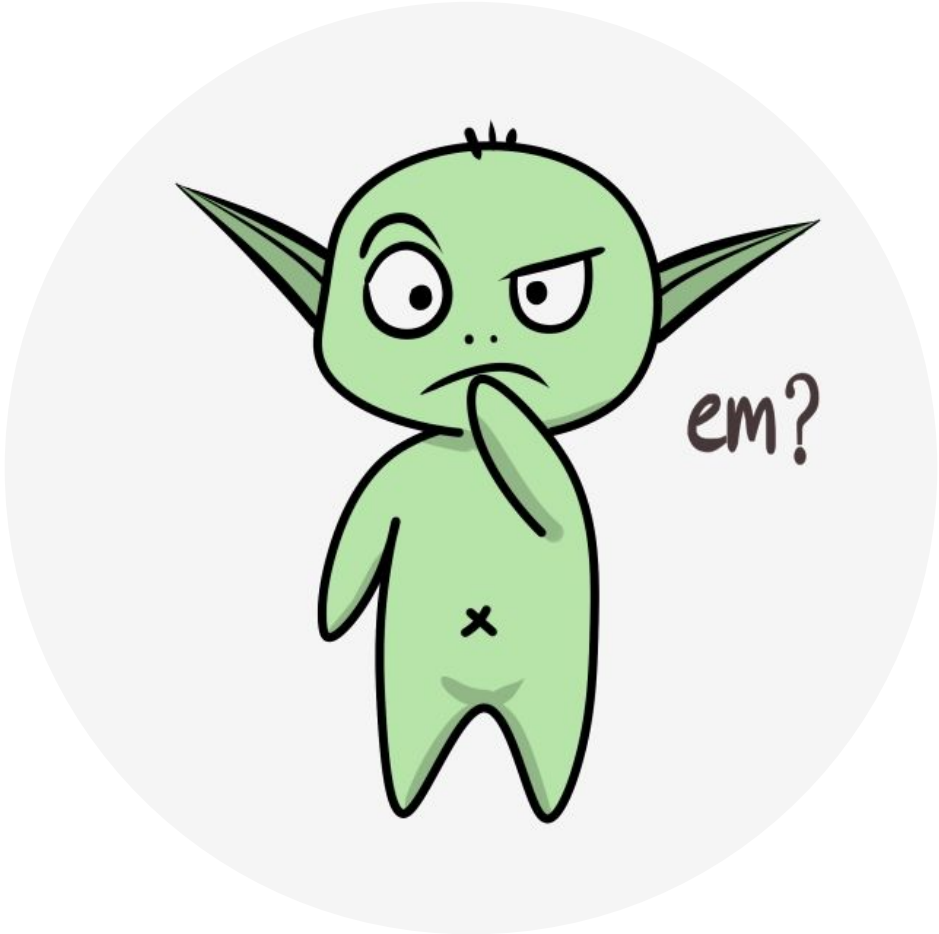
- When doing nonlinear function approximation, we lose pretty much every guarantee we had (often, even convergence guarantees).



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

Question

How are tabular methods related to linear function approximation?



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

Stochastic-gradient Methods

- The approximate value function, $\hat{v}(s,\mathbf{w})$, needs to be a differentiable function of $\mathbf{w}$ for all states.
- For this class, consider that, on each step, we observe a new example $S_t \mapsto v_\pi(S_t)$. Even with the exact target, we need to properly allocate resources.
- *Stochastic gradient-descent (SGD)* is a great strategy:

$$\begin{aligned}\mathbf{w}_{t+1} &\doteq \mathbf{w}_t - \frac{1}{2}\alpha\nabla\Big[v_\pi(S_t) - \hat{v}(S_t,\mathbf{w}_t)\Big]^2 \\ &= \mathbf{w}_t + \alpha\Big[v_\pi(S_t) - \hat{v}(S_t,\mathbf{w}_t)\Big]\nabla\hat{v}(S_t,\mathbf{w}_t)\end{aligned}$$

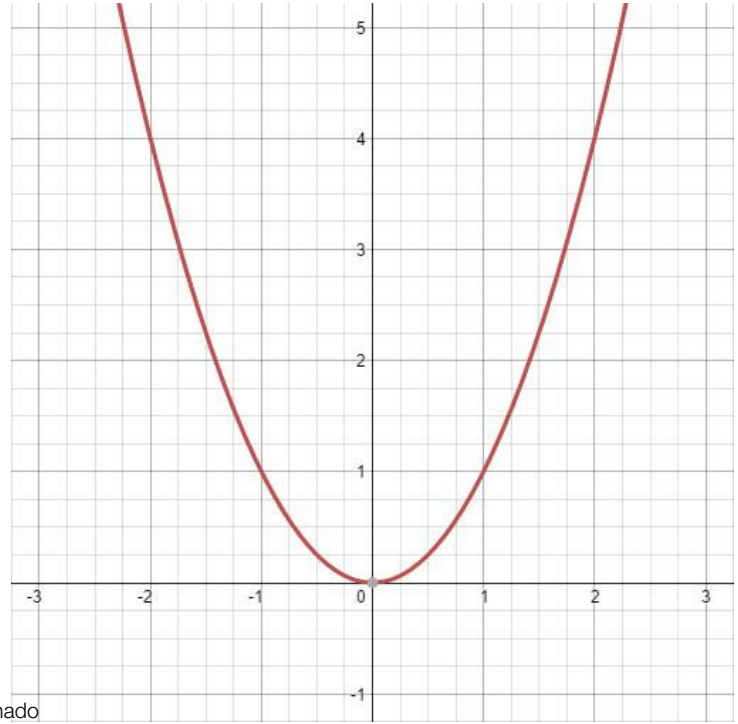
$$\nabla f(\mathbf{w}) \doteq \left(\frac{\partial f(\mathbf{w})}{\partial w_1}, \frac{\partial f(\mathbf{w})}{\partial w_2}, \ldots, \frac{\partial f(\mathbf{w})}{\partial w_d}\right)^\top$$

Few (one) state at a time

We need to consider the impact of our update. Thus, small updates are often preferred.

Example – Stochastic gradient descent

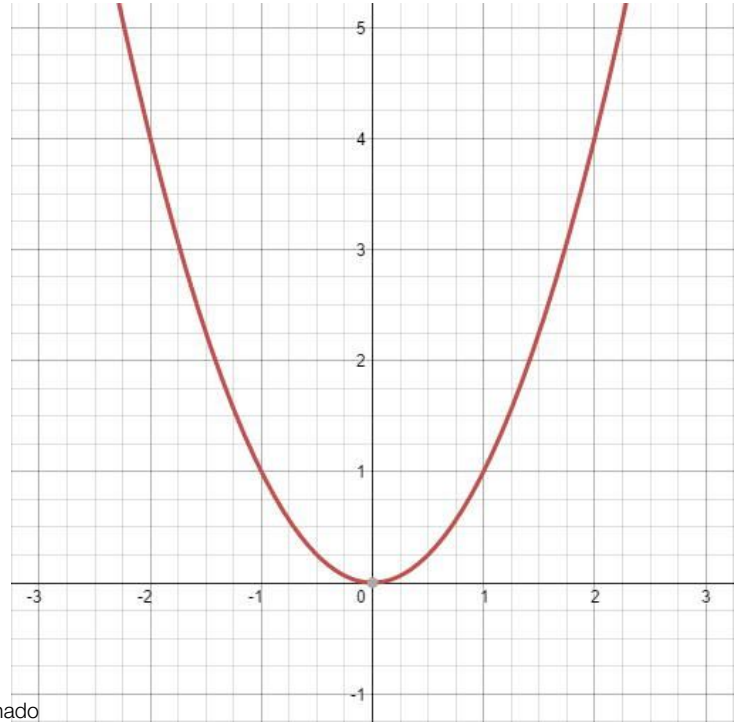
Say we have a function $f(z) = z^2$, and we want to find the z that minimizes its value.



$$z' \leftarrow z \pm \alpha \nabla_z f(z)$$

Example – Stochastic gradient descent

Say we have a function $f(z) = z^2$, and we want to find the z that minimizes its value.

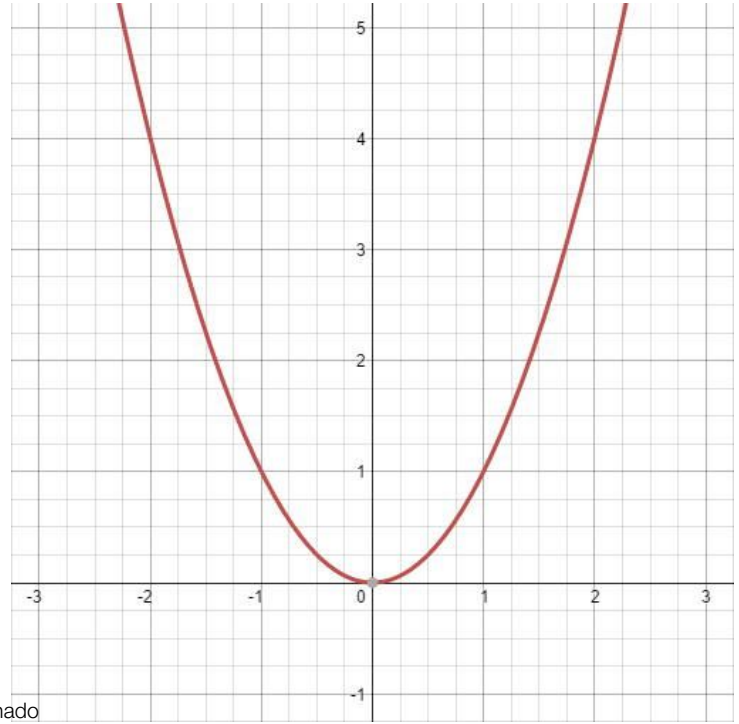


$$\frac{df(z)}{dz} =$$

$$z' \leftarrow z \pm \alpha \nabla_z f(z)$$

Example – Stochastic gradient descent

Say we have a function $f(z) = z^2$, and we want to find the z that minimizes its value.

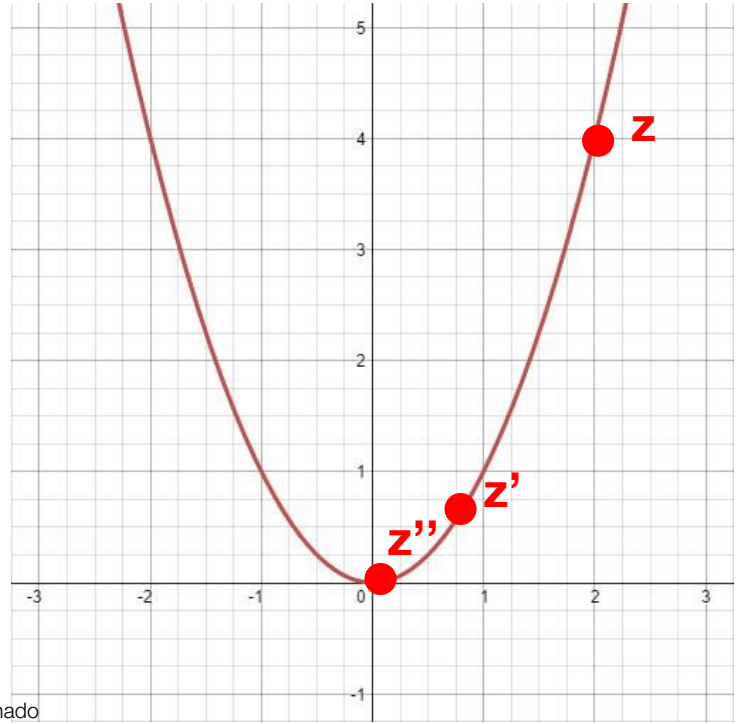


$$\frac{df(z)}{dz} = 2z$$

$$z' \leftarrow z \pm \alpha \nabla_z f(z)$$

Example – Stochastic gradient descent (intuition)

Say we have a function $f(z) = z^2$, and we want to find the z that minimizes its value.



$$\frac{df(z)}{dz} = 2z$$

$$z' \leftarrow z \pm \alpha \nabla_z f(z)$$

$$\alpha = 0.4$$

$\nabla f(4) = 2 \times 4 = 8$
$z' \leftarrow 4 - 0.4 \times 8$
$z' = 0.8$

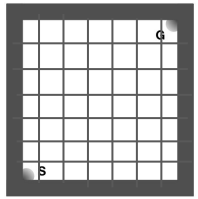
$\nabla f(0.8) = 2 \times 0.8 = 1.6$
$z'' \leftarrow 0.8 - 0.4 \times 1.6$
$z'' = 0.16$

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`

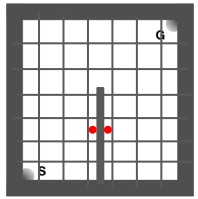
1. Specify a FA architecture (parametric form of v_π)

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



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

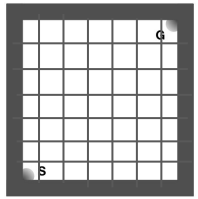
- We will use *state aggregation with linear function approximation*
- State aggregation
  - The features are always binary with only a single active feature that is not zero



State aggregation is far from perfect!

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

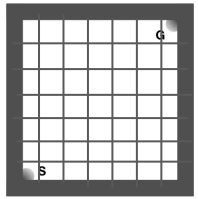
- We will use *state aggregation* with *linear function approximation*
- State aggregation
  - The features are always binary with only a single active feature that is not zero
- Value function
  - Linear function



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

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

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

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

A More Realistic Update

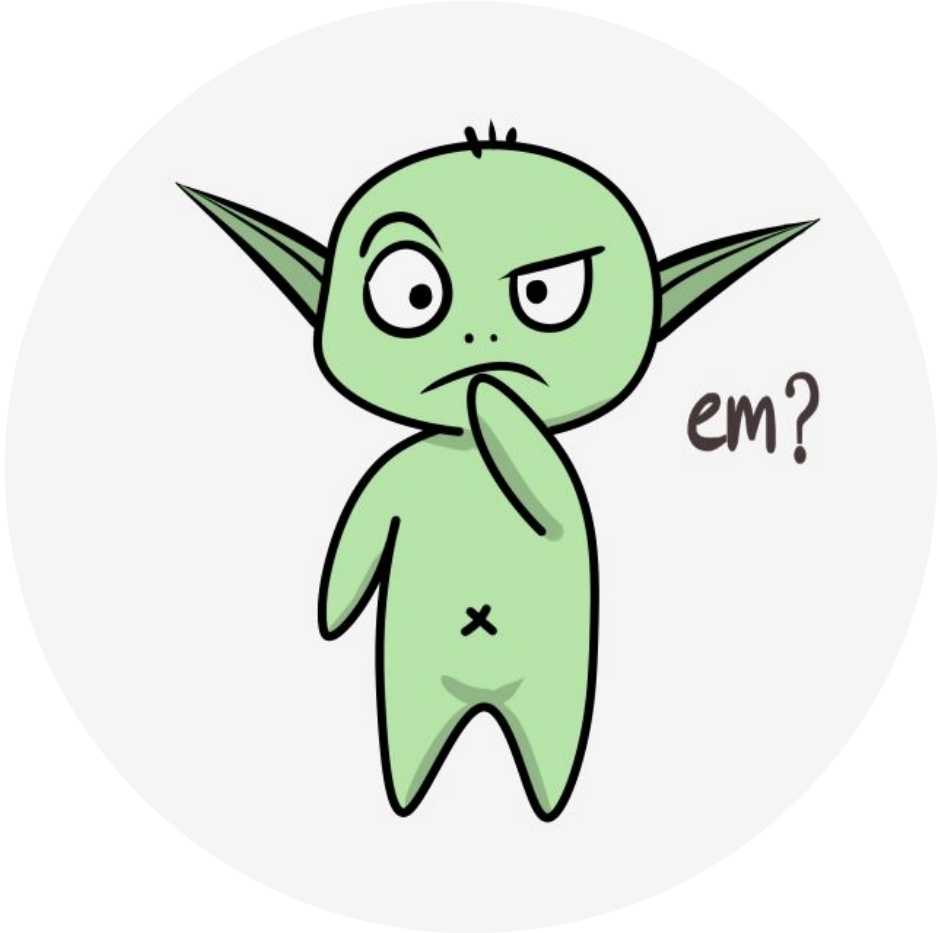
- Let U_t denote the *t*-th training example, $S_t \mapsto v_\pi(S_t)$, of some (possibly random), approximation to the true value.

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

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, $t = 0, 1, \ldots, T-1$:
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha \big[G_t - \hat{v}(S_t, \mathbf{w}) \big] \nabla \hat{v}(S_t, \mathbf{w})$

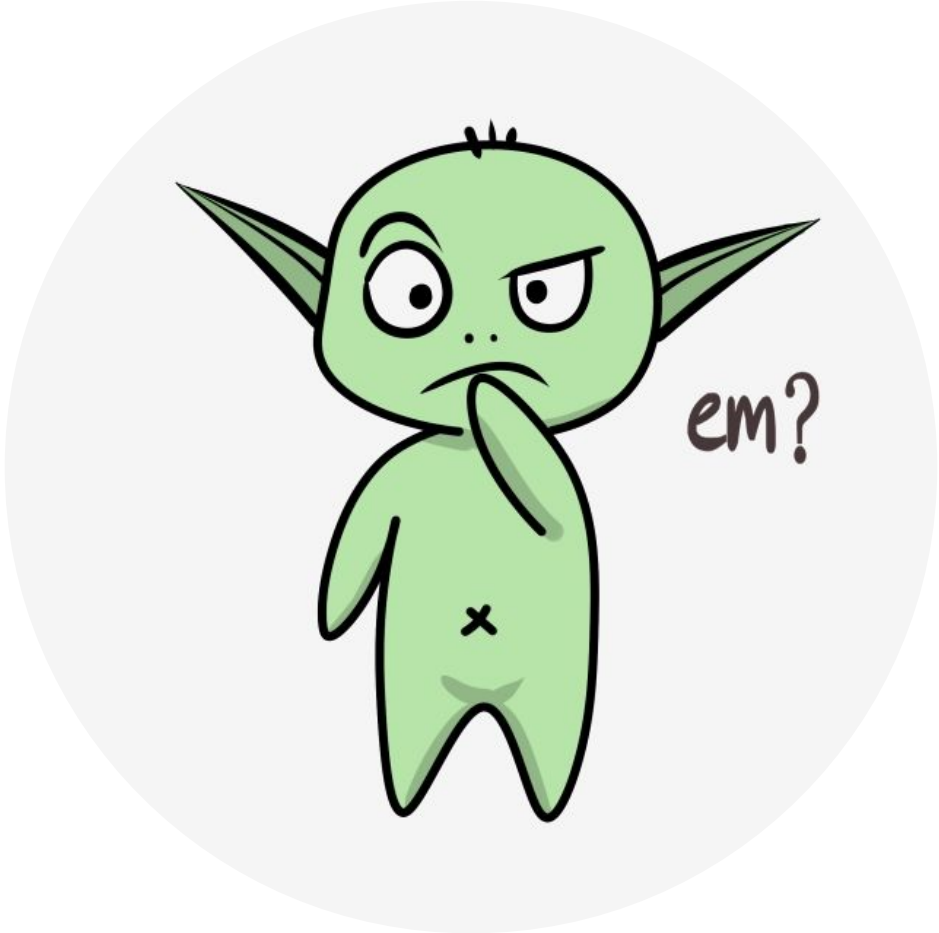


https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

A Clearer Instantiation — Linear Function Approximation

- Let $\hat{v}(\mathbf{x}, \mathbf{w}) = \mathbf{x}^\top \mathbf{w}$. We have $\nabla_{\mathbf{w}} \hat{v}(\mathbf{x}, \mathbf{w}) = \mathbf{x}(s)$.
- Thus, $\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha [U_t - \hat{v}(\mathbf{x}, \mathbf{w})] \nabla_{\mathbf{w}} \hat{v}(\mathbf{x}, \mathbf{w})$ becomes:

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



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

Semi-gradient TD

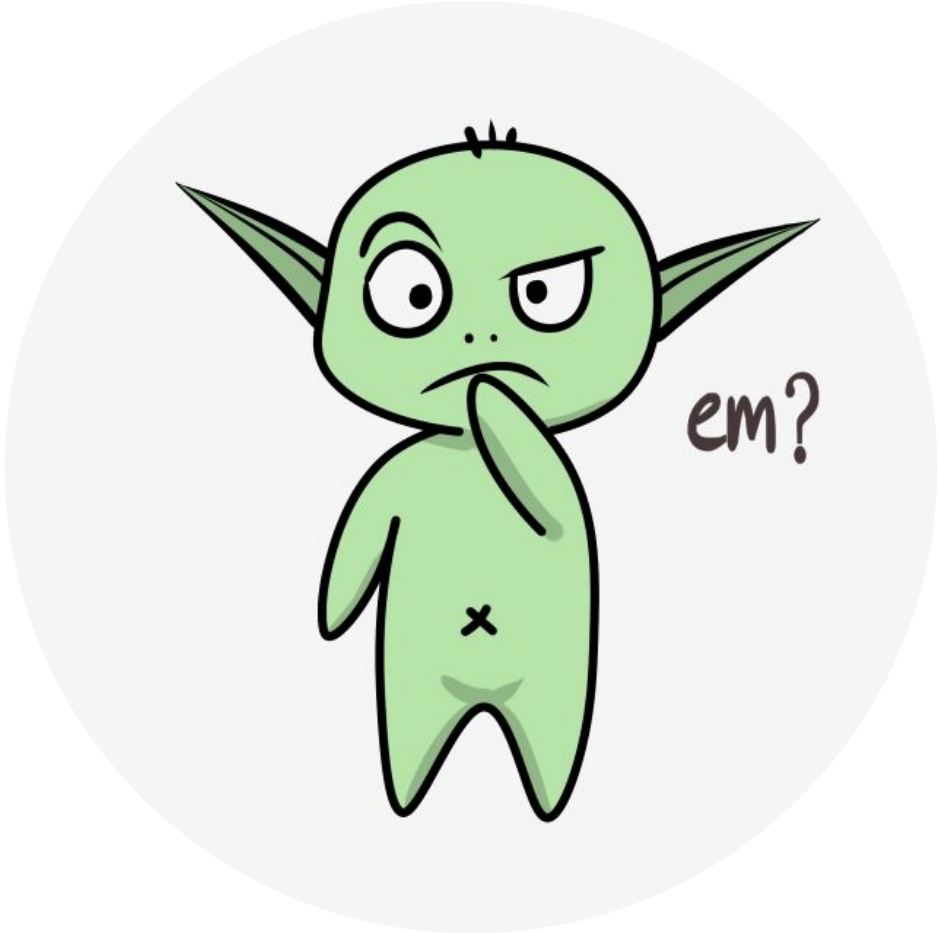
- What if $U_t \doteq R_{t+1} + \gamma\, \hat{v}(S_{t+1}, \mathbf{w}_t)$?
- We lose several guarantees when we use a bootstrapping estimate as target.
  - The target now also depends on the value of $\mathbf{w}_t$, so the target is not independent of $\mathbf{w}_t$.
- Bootstrapping are not instances 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(0)

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

TD Fixed Point with Linear Function Approximation

- We do have convergence results for linear function approximation.

$$\begin{aligned}\mathbf{w}_{t+1} &\doteq \mathbf{w}_t + \alpha\Big(R_{t+1} + \gamma \mathbf{w}_t^\top \mathbf{x}_{t+1} - \mathbf{w}_t^\top \mathbf{x}_t\Big)\mathbf{x}_t \\ &= \mathbf{w}_t + \alpha\Big(R_{t+1}\mathbf{x}_t - \mathbf{x}_t\big(\mathbf{x}_t - \gamma \mathbf{x}_{t+1}\big)^\top \mathbf{w}_t\Big)\end{aligned}$$

TD Fixed Point with Linear Function Approximation

- We do have convergence results for linear function approximation.

$$\begin{aligned}\mathbf{w}_{t+1} &\doteq \mathbf{w}_t + \alpha\Big(R_{t+1} + \gamma \mathbf{w}_t^\top \mathbf{x}_{t+1} - \mathbf{w}_t^\top \mathbf{x}_t\Big)\mathbf{x}_t \\ &= \mathbf{w}_t + \alpha\Big(R_{t+1}\mathbf{x}_t - \mathbf{x}_t\big(\mathbf{x}_t - \gamma \mathbf{x}_{t+1}\big)^\top \mathbf{w}_t\Big)\end{aligned}$$

In a steady state, for any given $\mathbf{w}_t$, the expected next weight vector can be written

$$\mathbb{E}[\mathbf{w}_{t+1}|\mathbf{w}_t] = \mathbf{w}_t + \alpha(\mathbf{b} - \mathbf{A}\mathbf{w}_t)$$

$$\text{where} \quad \mathbf{b} \doteq \mathbb{E}[R_{t+1}\mathbf{x}_t] \in \mathbb{R}^d \quad \text{and} \quad \mathbf{A} \doteq \mathbb{E}\Big[\mathbf{x}_t\big(\mathbf{x}_t - \gamma \mathbf{x}_{t+1}\big)^\top\Big] \in \mathbb{R}^{d\times d}$$

TD Fixed Point with Linear Function Approximation

- We do have convergence results for linear function approximation.

$$\begin{aligned}\mathbf{w}_{t+1} &\doteq \mathbf{w}_t + \alpha\Big(R_{t+1} + \gamma \mathbf{w}_t^\top \mathbf{x}_{t+1} - \mathbf{w}_t^\top \mathbf{x}_t\Big)\mathbf{x}_t \\ &= \mathbf{w}_t + \alpha\Big(R_{t+1}\mathbf{x}_t - \mathbf{x}_t\big(\mathbf{x}_t - \gamma \mathbf{x}_{t+1}\big)^\top \mathbf{w}_t\Big)\end{aligned}$$

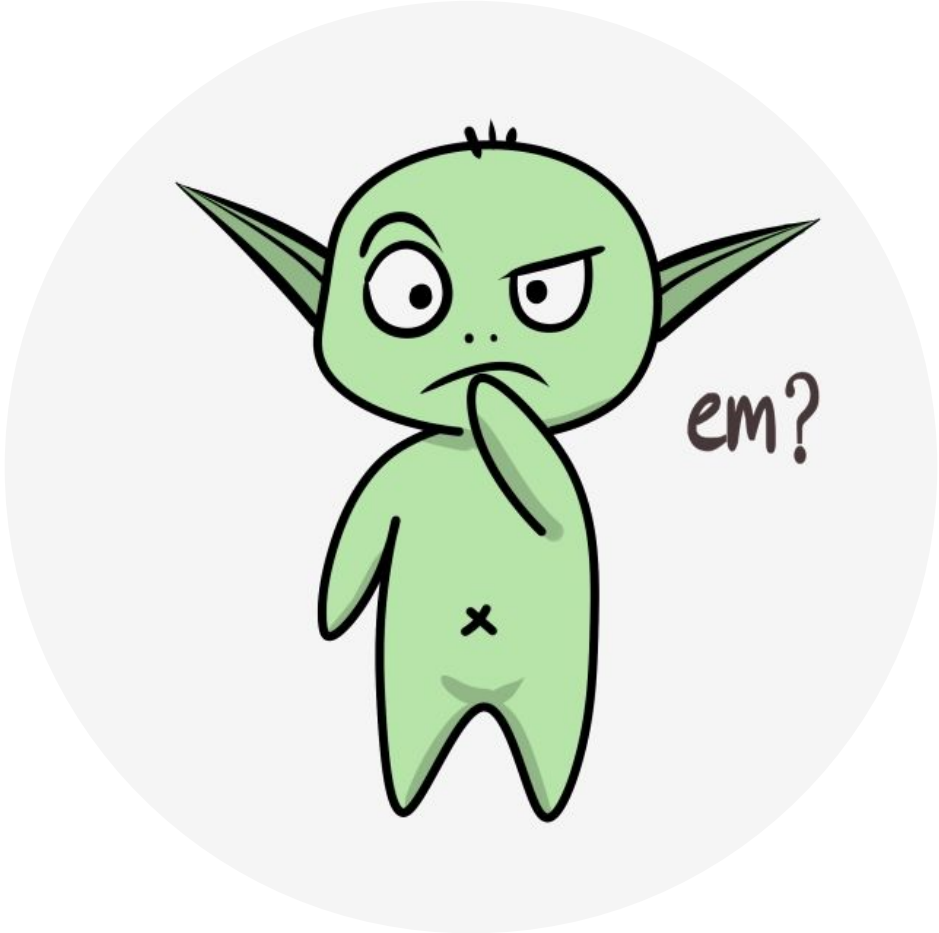
In a steady state, for any given $\mathbf{w}_t$, the expected next weight vector can be written

$$\mathbb{E}[\mathbf{w}_{t+1}|\mathbf{w}_t] = \mathbf{w}_t + \alpha(\mathbf{b} - \mathbf{A}\mathbf{w}_t)$$

$$\text{where} \quad \mathbf{b} \doteq \mathbb{E}[R_{t+1}\mathbf{x}_t] \in \mathbb{R}^d \quad \text{and} \quad \mathbf{A} \doteq \mathbb{E}\Big[\mathbf{x}_t\big(\mathbf{x}_t - \gamma \mathbf{x}_{t+1}\big)^\top\Big] \in \mathbb{R}^{d\times d}$$

It converges to:

$$\begin{aligned}&& \mathbf{b} - \mathbf{A}\mathbf{w}_{\text{TD}} &= \mathbf{0} \\ \Rightarrow && \mathbf{b} &= \mathbf{A}\mathbf{w}_{\text{TD}} \\ \Rightarrow && \mathbf{w}_{\text{TD}} &\doteq \mathbf{A}^{-1}\mathbf{b}.\end{aligned}$$



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

Example / Exercise

Let $\hat{v}(\mathbf{x}, \mathbf{w}) = \mathbf{x}^\top \mathbf{w}$, and consider the update rule: $\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha [G_t - \mathbf{x}_t^\top \mathbf{w}_t]\mathbf{x}_t$.
Let $\alpha = 0.1$, and consider $\mathbf{w}_5 = [1.0, 0.5, 3.0]^\top$ and $\mathbf{x}_5 = [0, 2, -1]^\top$.

What's $\hat{v}(\mathbf{x}_5, \mathbf{w}_5)$? What's $\mathbf{w}_6$ when applying the update rule above for $G_5 = 10$?
What do we observe from this process?

Example / Exercise

Example / Exercise

Let $\hat{v}(\mathbf{x}, \mathbf{w}) = \mathbf{x}^\top \mathbf{w}$, and consider the update rule: $\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\,[G_t - \mathbf{x}_t^\top \mathbf{w}_t]\mathbf{x}_t$.
Let $\alpha = 0.1$, and consider $\mathbf{w}_5 = [1.0, 0.5, 3.0]^\top$ and $\mathbf{x}_5 = [0, 2, -1]^\top$.

What's $\hat{v}(\mathbf{x}_5, \mathbf{w}_5)$? What's $\mathbf{w}_6$ when applying the update rule above for $G_5 = 10$? What do we observe from this process?

$$\hat{v}(\mathbf{x}_5, \mathbf{w}_5) = \mathbf{x}_5^\top \mathbf{w}_5 = \begin{pmatrix} 0, 2, -1 \end{pmatrix} \begin{pmatrix} 1.0 \\ 0.5 \\ 3.0 \end{pmatrix} = 0 \times 1.0 + 2 \times 0.5 - 1 \times 3.0 = 0 + 1 - 3 = -2$$

Example / Exercise

Let $\hat{v}(\mathbf{x}, \mathbf{w}) = \mathbf{x}^\top \mathbf{w}$, and consider the update rule: $\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\,[G_t - \mathbf{x}_t^\top \mathbf{w}_t]\mathbf{x}_t$.
Let $\alpha = 0.1$, and consider $\mathbf{w}_5 = [1.0, 0.5, 3.0]^\top$ and $\mathbf{x}_5 = [0, 2, -1]^\top$.

What's $\hat{v}(\mathbf{x}_5, \mathbf{w}_5)$? What's $\mathbf{w}_6$ when applying the update rule above for $G_5 = 10$? What do we observe from this process?

$\hat{v}(\mathbf{x}_5, \mathbf{w}_5) = -2$

$$\mathbf{w}_6 \leftarrow \mathbf{w}_5 + \alpha\,[G_5 - \hat{v}(\mathbf{x}_5, \mathbf{w}_5)]\mathbf{x}_5$$

$$\mathbf{w}_6 \leftarrow \begin{pmatrix} 1.0 \\ 0.5 \\ 3.0 \end{pmatrix} + 0.1\,[10 - -2] \begin{pmatrix} 0 \\ 2 \\ -1 \end{pmatrix}$$

Example / Exercise

Let $\hat{v}(\mathbf{x}, \mathbf{w}) = \mathbf{x}^\top \mathbf{w}$, and consider the update rule: $\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\,[G_t - \mathbf{x}_t^\top \mathbf{w}_t]\mathbf{x}_t$.
Let $\alpha = 0.1$, and consider $\mathbf{w}_5 = [1.0, 0.5, 3.0]^\top$ and $\mathbf{x}_5 = [0, 2, -1]^\top$.

What's $\hat{v}(\mathbf{x}_5, \mathbf{w}_5)$? What's $\mathbf{w}_6$ when applying the update rule above for $G_5 = 10$? What do we observe from this process?

$\hat{v}(\mathbf{x}_5, \mathbf{w}_5) = -2$

$$\mathbf{w}_6 \leftarrow \mathbf{w}_5 + \alpha\,[G_5 - \hat{v}(\mathbf{x}_5, \mathbf{w}_5)]\mathbf{x}_5$$

$$\mathbf{w}_6 \leftarrow \begin{pmatrix}1.0\\0.5\\3.0\end{pmatrix} + 0.1\,[10 - -2]\begin{pmatrix}0\\2\\-1\end{pmatrix} \qquad \mathbf{w}_6 \leftarrow \begin{pmatrix}1.0\\0.5\\3.0\end{pmatrix} + 1.2\begin{pmatrix}0\\2\\-1\end{pmatrix}$$

Example / Exercise

Let $\hat{v}(\mathbf{x}, \mathbf{w}) = \mathbf{x}^\top \mathbf{w}$, and consider the update rule: $\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\,[G_t - \mathbf{x}_t^\top \mathbf{w}_t]\mathbf{x}_t$.
Let $\alpha = 0.1$, and consider $\mathbf{w}_5 = [1.0, 0.5, 3.0]^\top$ and $\mathbf{x}_5 = [0, 2, -1]^\top$.

What's $\hat{v}(\mathbf{x}_5, \mathbf{w}_5)$? What's $\mathbf{w}_6$ when applying the update rule above for $G_5 = 10$? What do we observe from this process?

$\hat{v}(\mathbf{x}_5, \mathbf{w}_5) = -2$

$$\mathbf{w}_6 \leftarrow \mathbf{w}_5 + \alpha\,[G_5 - \hat{v}(\mathbf{x}_5, \mathbf{w}_5)]\mathbf{x}_5$$

$$\mathbf{w}_6 \leftarrow \begin{pmatrix}1.0\\0.5\\3.0\end{pmatrix} + 0.1\,[10 - -2]\begin{pmatrix}0\\2\\-1\end{pmatrix} \qquad \mathbf{w}_6 \leftarrow \begin{pmatrix}1.0\\0.5\\3.0\end{pmatrix} + 1.2\begin{pmatrix}0\\2\\-1\end{pmatrix} \qquad \mathbf{w}_6 \leftarrow \begin{pmatrix}1.0\\0.5\\3.0\end{pmatrix} + \begin{pmatrix}0\\2.4\\-1.2\end{pmatrix}$$

Example / Exercise

Let $\hat{v}(\mathbf{x}, \mathbf{w}) = \mathbf{x}^\top \mathbf{w}$, and consider the update rule: $\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\,[G_t - \mathbf{x}_t^\top \mathbf{w}_t]\mathbf{x}_t$.
Let $\alpha = 0.1$, and consider $\mathbf{w}_5 = [1.0, 0.5, 3.0]^\top$ and $\mathbf{x}_5 = [0, 2, -1]^\top$.

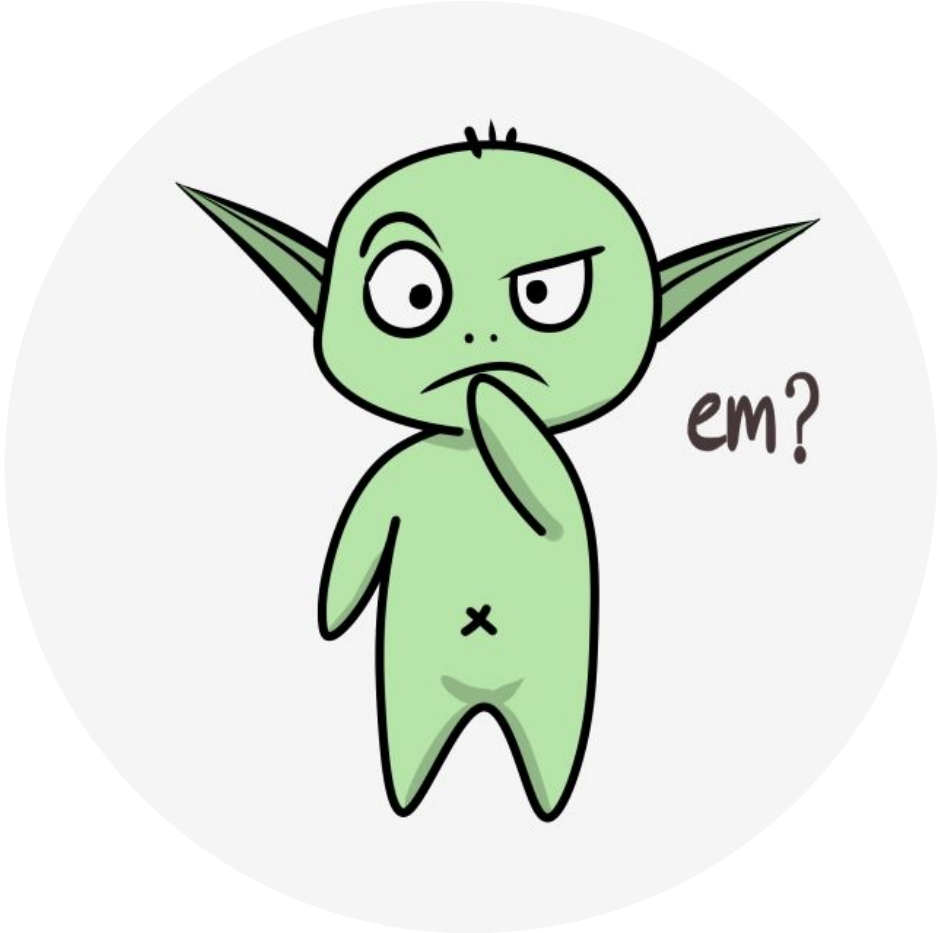
What's $\hat{v}(\mathbf{x}_5, \mathbf{w}_5)$? What's $\mathbf{w}_6$ when applying the update rule above for $G_5 = 10$? What do we observe from this process?

$\hat{v}(\mathbf{x}_5, \mathbf{w}_5) = -2$

$$\mathbf{w}_6 \leftarrow \mathbf{w}_5 + \alpha\,[G_5 - \hat{v}(\mathbf{x}_5, \mathbf{w}_5)]\mathbf{x}_5$$

$$\mathbf{w}_6 \leftarrow \begin{pmatrix}1.0\\0.5\\3.0\end{pmatrix} + 0.1\,[10 - -2]\begin{pmatrix}0\\2\\-1\end{pmatrix} \qquad \mathbf{w}_6 \leftarrow \begin{pmatrix}1.0\\0.5\\3.0\end{pmatrix} + 1.2\begin{pmatrix}0\\2\\-1\end{pmatrix} \qquad \mathbf{w}_6 \leftarrow \begin{pmatrix}1.0\\0.5\\3.0\end{pmatrix} + \begin{pmatrix}0\\2.4\\-1.2\end{pmatrix}$$

$$\mathbf{w}_6 = \begin{pmatrix}1.0, 2.9, 1.8\end{pmatrix}^\top$$



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

Example / Exercise

Question 4. Consider an episodic MDP with the state set $\mathcal{S} = \{1, 2, 3\}$ and a discount rate $\gamma = 0.9$. Suppose $\hat{v}(s; w) = w \cdot s$, where $w \in \mathbb{R}$ and $s \in \mathcal{S}$. The agent observes the following episodic data before reaching the terminal state at time $t = 3$, while following some fixed policy, where $\perp$ denotes the terminal state:

$$S_0 = 1, R_1 = -7, S_1 = 3, R_2 = 5, S_2 = 1, R_3 = 10, S_3 = \perp.$$

Assume that w is initialized to 0 at the beginning of learning and the step-size is fixed to $\alpha = 0.1$. Then

(a) What is $\hat{v}(1; w)$ when w is updated using MOnte-Carlo on this data?

(b) What is $\hat{v}(1; w)$ when w is updated using semi-gradient linear TD on this data?

(c) Again, compute the estimate $\hat{v}(1; w)$ for both gradient Monte-Carlo and semi-gradient Linear TD, when w is updated after seeing the same episode using the respective algorithms.

(Write down the numeric result to two decimal places.)