"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

Class 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.

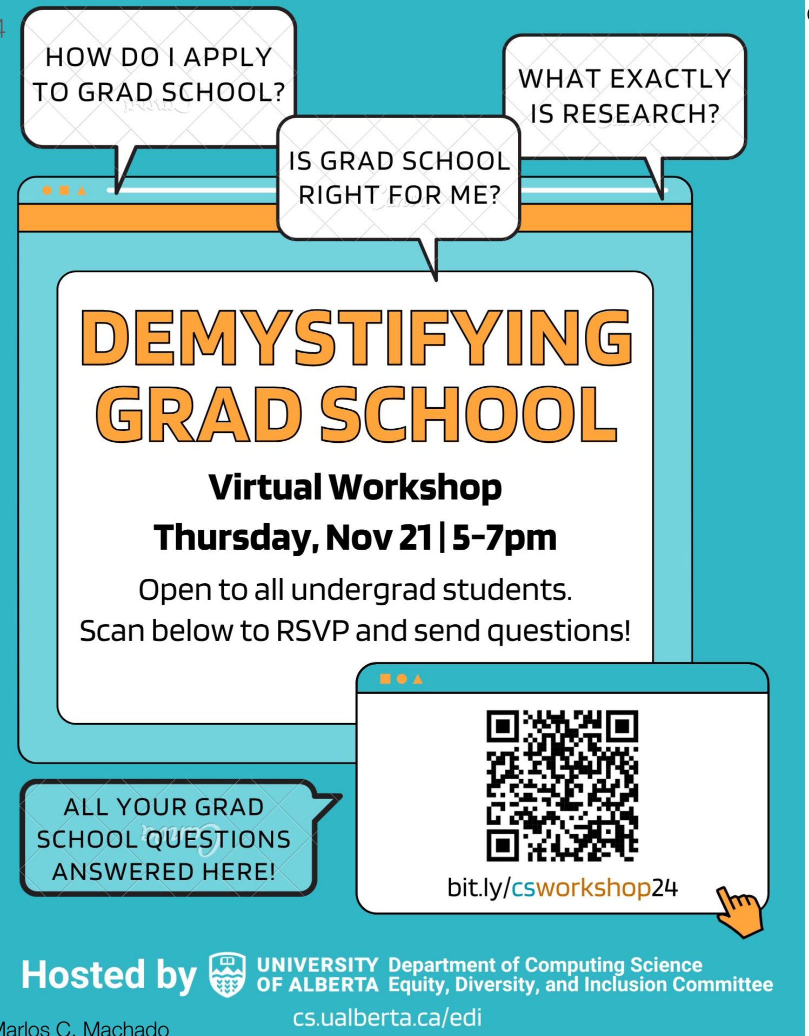
You **need** to **check**, **every time**, if you are in the private session and if you are submitting quizzes and assignments to the private section.

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

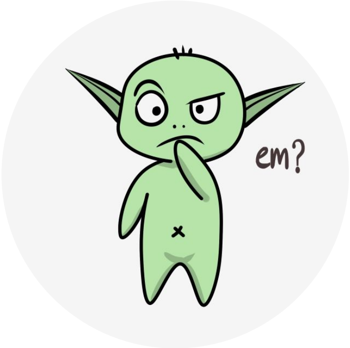
- The programming assignment is due today.
- Midterm 2 grades are available.
  - Exam viewing will be today and tomorrow.
  - Wednesday from 10am to 1pm at CSC 3-49.
  - Thursday from 1pm to 4pm at CSC 3-50.
- Next week is reading week.
- Our final exam has now been confirmed. It will indeed be on December 17th, 1pm, at CCIS 1-440.



RSVP form (not required, but appreciated):
https://docs.google.com/forms/d/11odJJgO3kgJ_XFDg9vnEz4FABjlNKx7-iL6AkCJ67ZQ/edit

Direct link to the zoom:
https://ualberta-ca.zoom.us/j/93282952849?pwd=eqE7hm46hwMJS02EZoqjw5GOngtWkK.1

Please, interrupt me at any time!



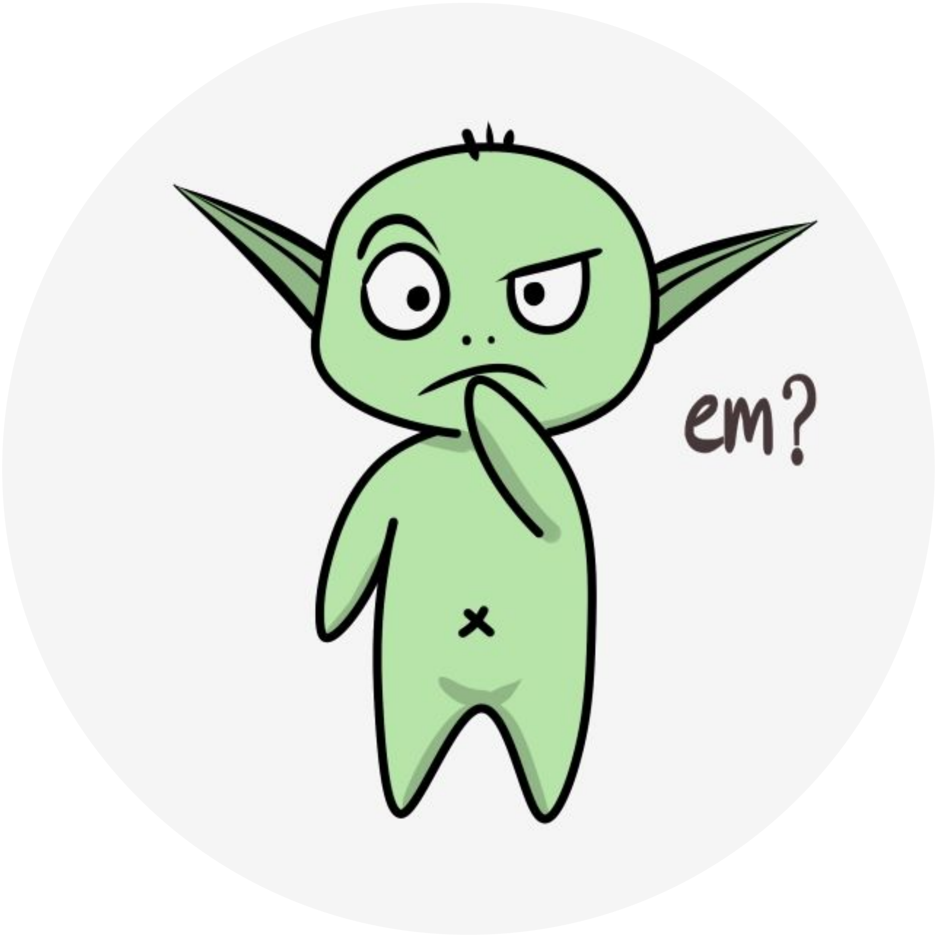
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Last Class: The Prediction Objective

- 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:

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

- 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

Stochastic-gradient Methods

- The approximate value function, $\hat{v}(s,\mathbf{w})$, needs to be a differentiable function of **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:

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

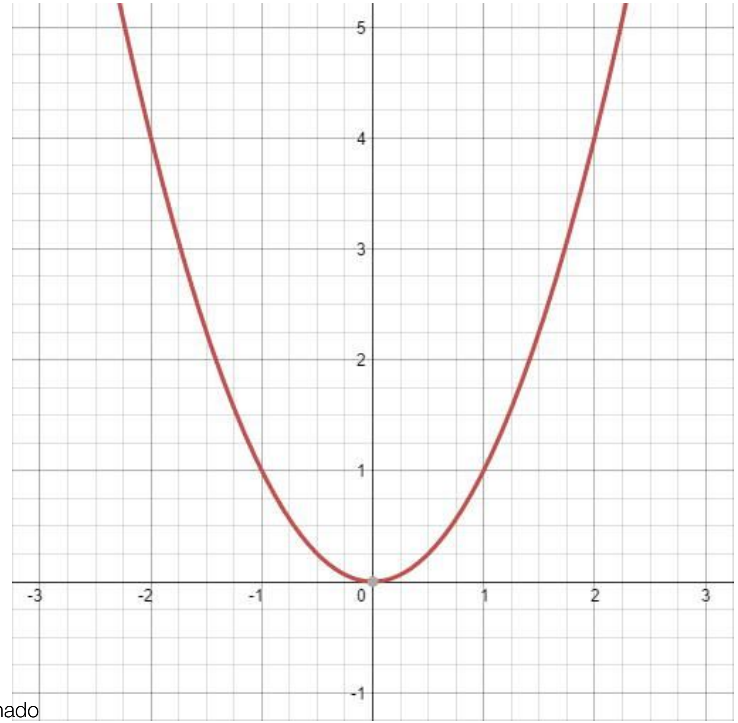
$$\nabla f(\mathbf{w}) \doteq \left(\frac{\partial f(\mathbf{w})}{\partial w_1}, \frac{\partial f(\mathbf{w})}{\partial w_2}, \ldots, \frac{\partial f(\mathbf{w})}{\partial w_d}\right)^\top$$

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

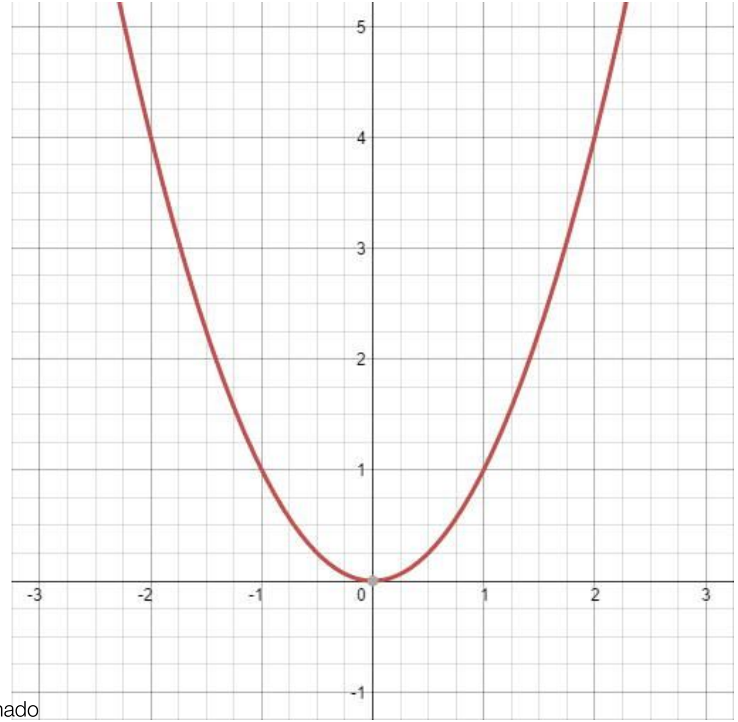
Say we have a function $f(z) = z^2$, and we want to find the z that minimizes its value.



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

Example – Stochastic gradient descent

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

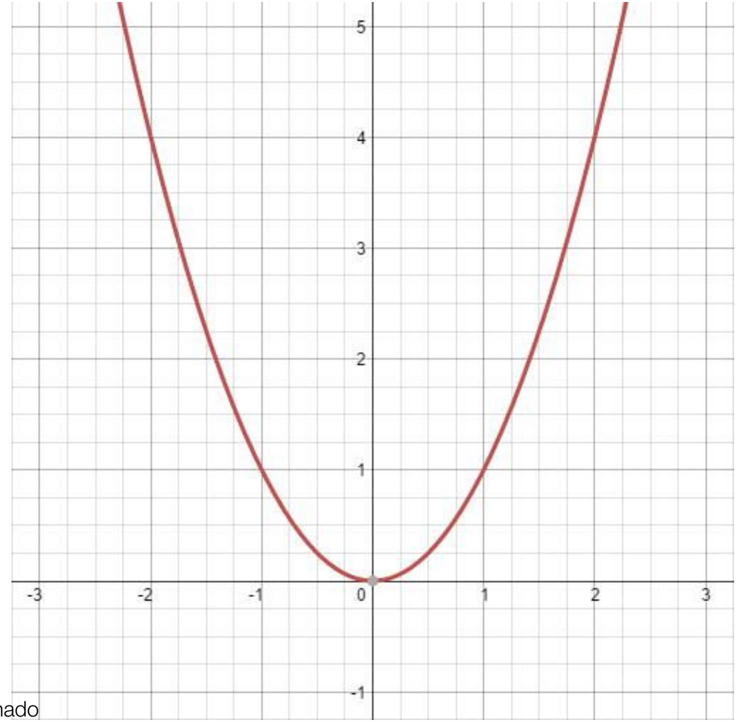


$$\frac{df(z)}{dz} =$$

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

Example – Stochastic gradient descent

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

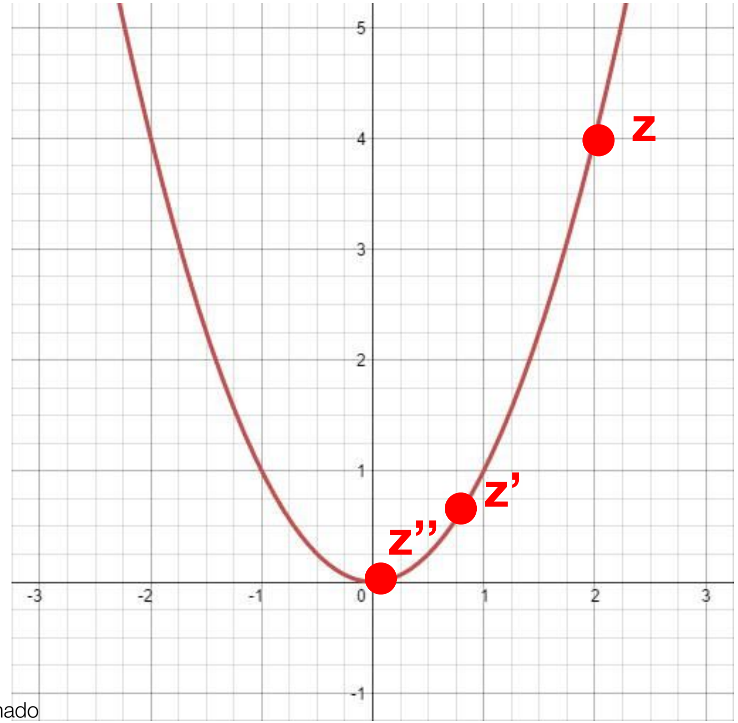


$$\frac{df(z)}{dz} = 2z$$

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

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.



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

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

$$\alpha = 0.4$$

$\nabla f(4) = 2 \times 4 = 8$

$z' \leftarrow 4 - 0.4 \times 8$

$z' = 0.8$

$\nabla f(0.8) = 2 \times 0.8 = 1.6$

$z'' \leftarrow 0.8 - 0.4 \times 1.6$

$z'' = 0.16$

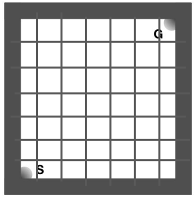
Recipe for Deriving a Concrete Algorithm for SGD

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

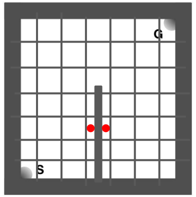
1. Specify a FA architecture (parametric form of v_π)

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



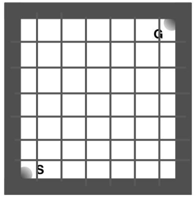
1. Specify a FA architecture (parametric form of v_π)

- 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)

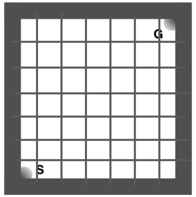


- We will use *state aggregation* with *linear function approximation*
- State aggregation
  - The features are always binary with only a single active feature that is not zero
- Value function
  - Linear function

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

2. Write down your objective function

- We will use the *value error*



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

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

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

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

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

5. Make a weight update rule

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

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

A More Realistic Update

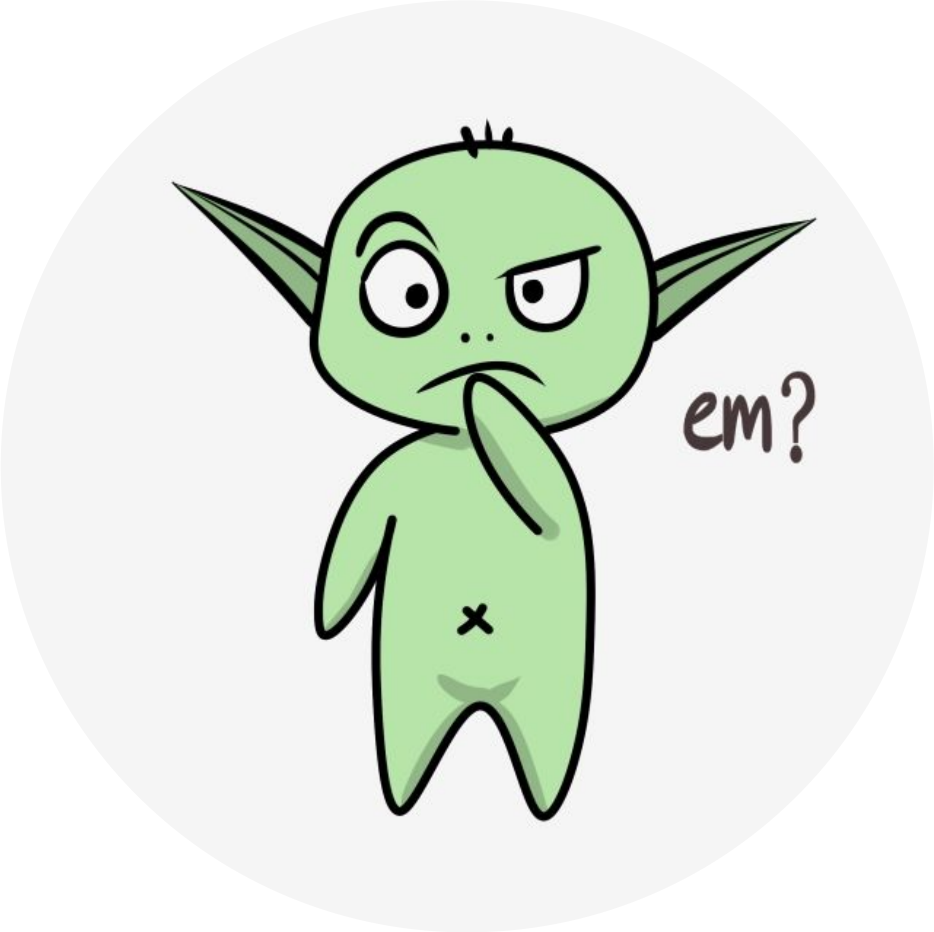
- Let U_t denote the *t*-th training example, $S_t \mapsto v_\pi(S_t)$, of some (possibly random), approximation to the true value.

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

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