
“The test of a man isn’t what you think he’ll do. It’s what he actually does.”
Frank Herbert, *Dune*
CMPUT 365
Introduction to RL
Marlos C. Machado
https://openart.ai/discovery/sd-1006656316309258270
Class 32/35

# 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.

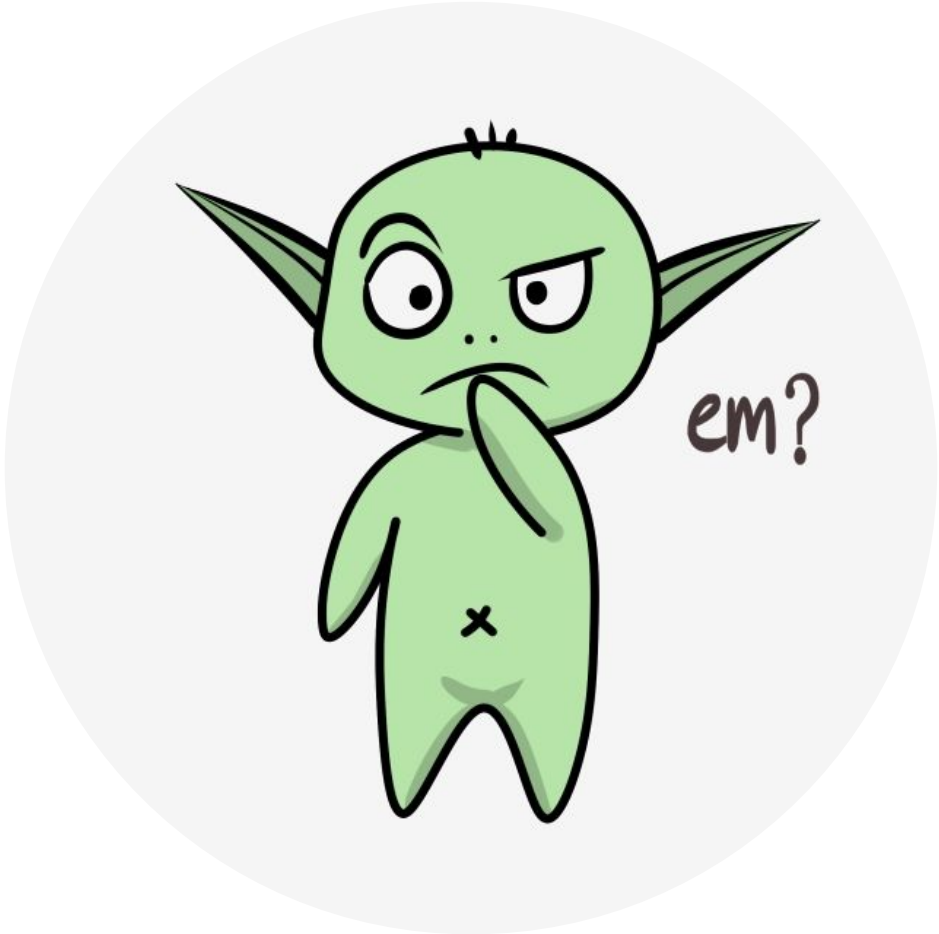
You **need** to **check**, **every time**, if you are in the private session and if you are submitting quizzes and assignments to the private section.

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

- I need to make adjustments for my office hours in the next two weeks (still same day):
  - This week, my office hours will be **shorter**: 2pm to 3pm
  - Next week, it will be **shifted** to 12:30pm to 2:30pm
- I gave you an **extension** for the last quiz of the course. Both the last quiz and the last programming assignment are due on Wednesday.
- Rich Sutton will give a guest lecture Dec 9th, Monday. Spread the word.
- A note on the final exam:
  - The required reading from the syllabus does not mean that's what will be covered in the final exam. There are some mismatches. Anything we discussed in class is fair game, including Maximization Bias and Double Learning (Sec. 6.7), and Nonlinear Function Approximation: Artificial Neural Networks (Sec. 9.7).
  - Final will be *2 hours long*, and questions will cover the whole term.
- SPOT Survey is still available for you.






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

# Chapter 13

# Policy Gradient Methods

# Policy Gradient Methods

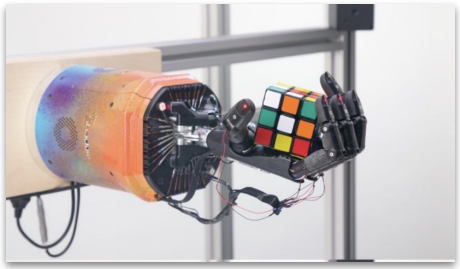
- Pretty much everything so far has been about action-value methods.
  - They learn value functions and then select an action based on their estimated action values.
- What if we learned the policy directly?

# Policy Gradient Methods– Why?

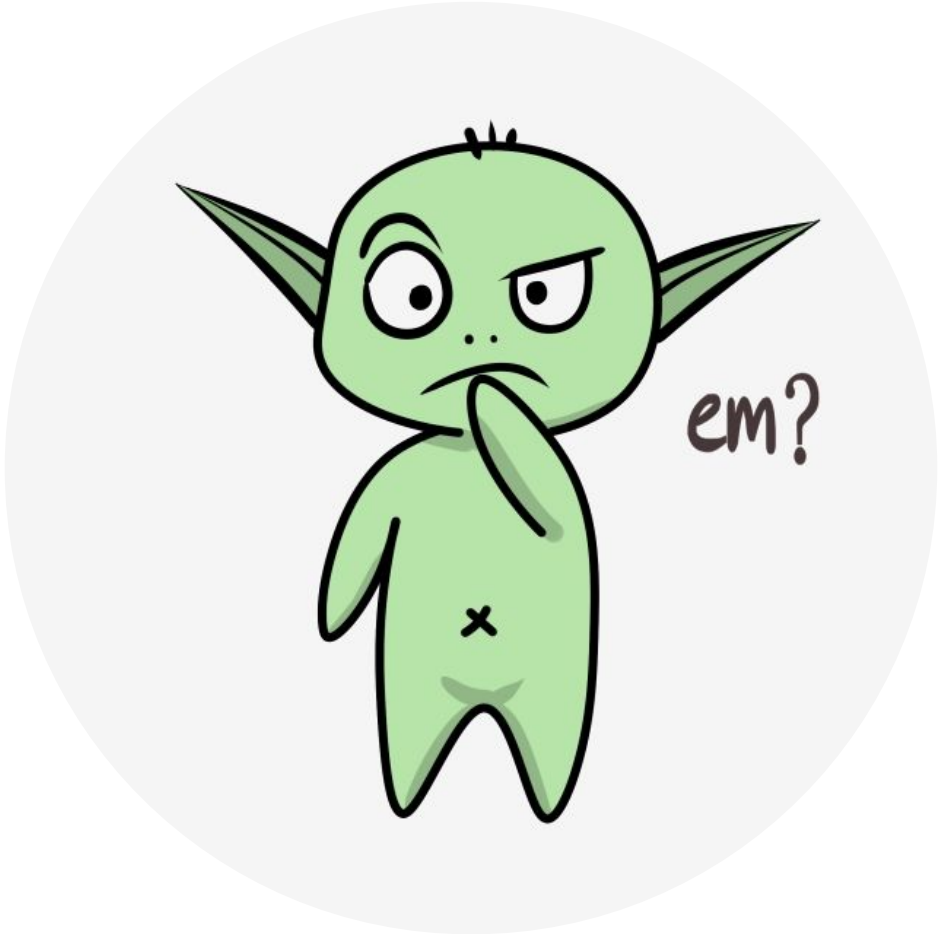
- Everything we discussed so far assumed we had a discrete action set.
  - Many problems we care about have an action set with continuous actions (e.g., torque in a motor).
- It naturally “scales” to function approximation and sometimes the PG algorithms have much nicer guarantees.
- Maybe one should directly optimize what they care about, which is the policy.
- It works and it is used everywhere now ¯_(ツ)_/¯

LLMs (e.g., ChatGPT)

Video games
(e.g., Dota2, StarCraft 2)



Robotics!





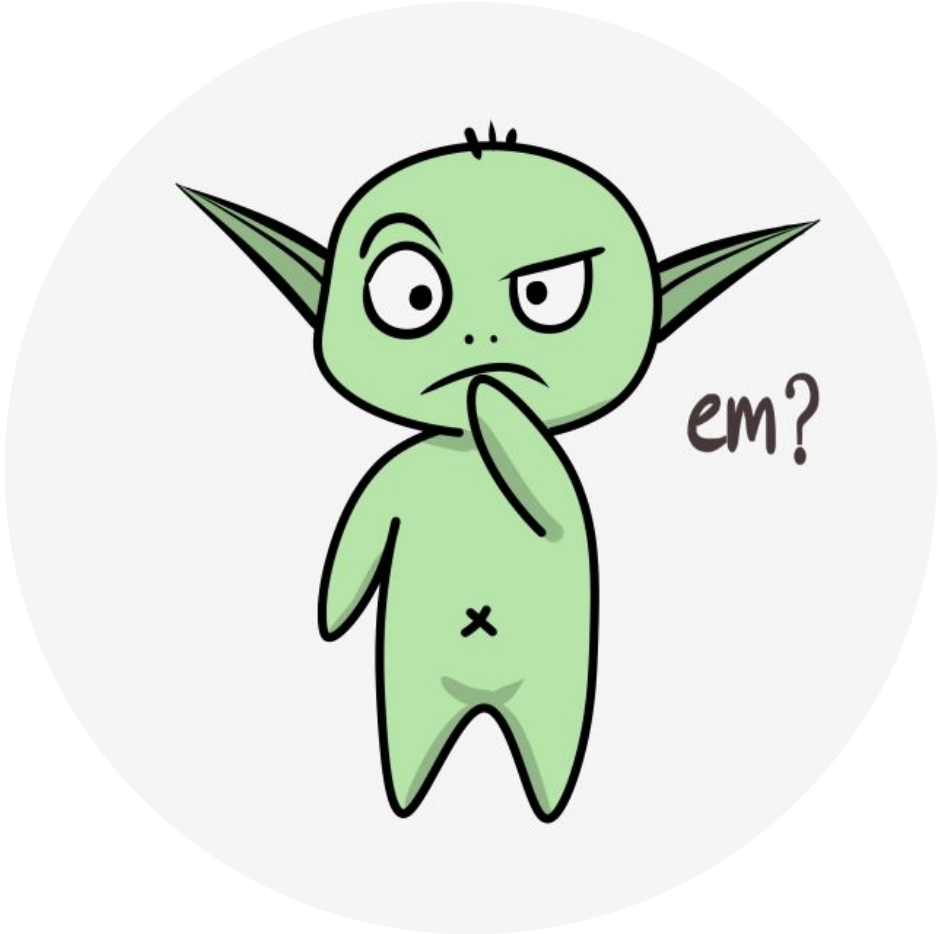

https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

# More Specifically

- Let $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ denote the policy's parameter vector. We then write: $\pi(a|s, \boldsymbol{\theta}) = \Pr(A_t = a \mid S_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta})$.

- We consider some scalar performance measure $J(\boldsymbol{\theta})$ with respect to the policy parameter. We perform gradient *ascent* in J:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \widehat{\nabla J(\boldsymbol{\theta}_t)}$$






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

# Gradient Bandit Algorithms (Chapter 2) – No parameterization

- Bandits! But instead of learning an estimate of the expected reward from each arm, we learn a numerical *preference* for each action a, denoted $H_t(a)$.
  - The larger the preference the more likely you are to take that action, but the preference has no additional semantics in terms of rewards.
  - If we offset all action preferences we will still select actions with the same probability.

- We choose actions according to a *softmax distribution (*i.e., Gibbs or Boltzmann distribution):

At first, all action preferences are the same

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

Notice we are not conditioning on *s*, because it is a bandits problem

# Some more on the Softmax

$$\frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}}$$

| | $a_1$ | $a_2$ | $a_3$ |
|---|---|---|---|
| **H(·)** | 1 | 1 | 1 |
| **π(·)** | 0.33 | 0.33 | 0.33 |

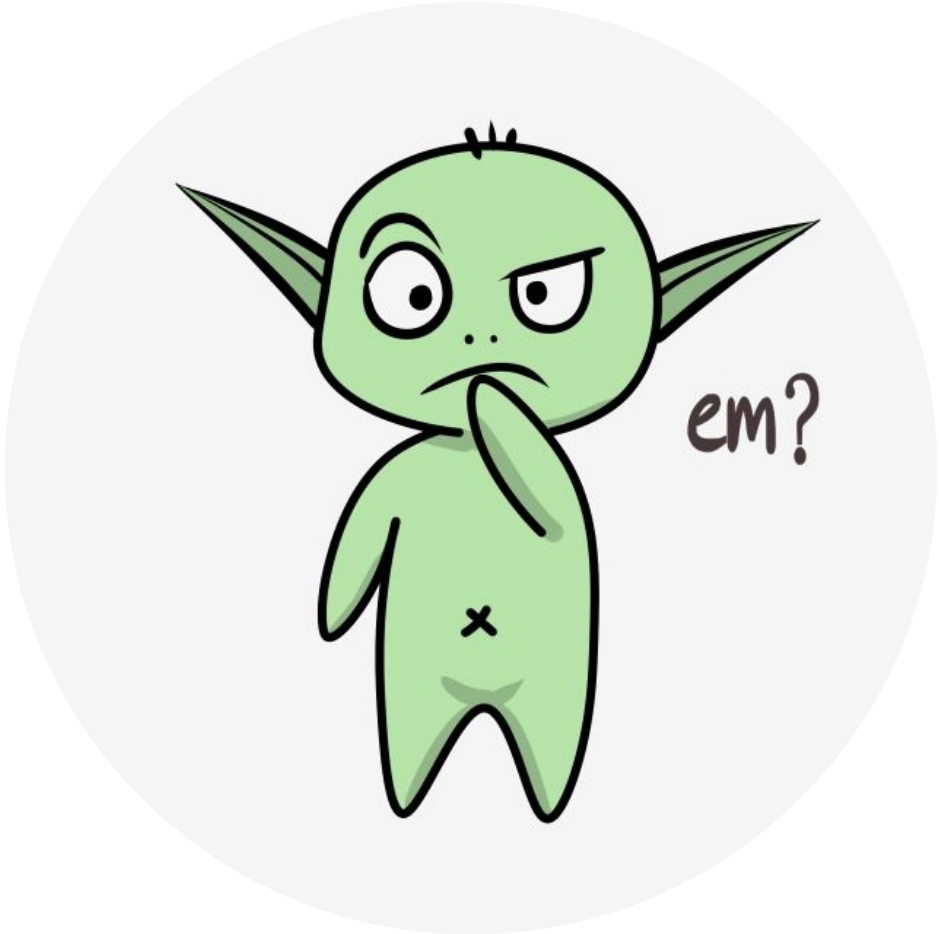
| | $a_1$ | $a_2$ | $a_3$ |
|---|---|---|---|
| **H(·)** | 4 | 1 | 1 |
| **π(·)** | 0.91 | 0.05 | 0.05 |

| | $a_1$ | $a_2$ | $a_3$ |
|---|---|---|---|
| **H(·)** | 3 | 2 | 1 |
| **π(·)** | 0.67 | 0.24 | 0.09 |

| | $a_1$ | $a_2$ | $a_3$ |
|---|---|---|---|
| **H(·)** | 3 | 1 | 1 |
| **π(·)** | 0.79 | 0.11 | 0.11 |

| | $a_1$ | $a_2$ | $a_3$ |
|---|---|---|---|
| **H(·)** | -4 | 1 | 1 |
| **π(·)** | 0 | 0.5 | 0.5 |

| | $a_1$ | $a_2$ | $a_3$ |
|---|---|---|---|
| **H(·)** | 4 | 3 | 2 |
| **π(·)** | 0.67 | 0.24 | 0.09 |






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

# Gradient Bandit Algorithms (Chapter 2) – No parameterization

- We change the preferences with stochastic gradient ascent.
- The idea:

$$H_{t+1}(a) \doteq H_t(a) + \alpha \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)}$$

where the measure of performance here is the expected reward:

$$\mathbb{E}[R_t] = \sum_x \pi_t(x) q_*(x)$$

Obviously, we don't know $q_*$.

… but we can be clever about it.

# Gradient Bandit Algorithms (Chapter 2) – Derivation

Let's look at the exact performance gradient:

$$\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \frac{\partial}{\partial H_t(a)} \left[ \sum_x \pi_t(x) q_*(x) \right]$$

$$= \sum_x q_*(x) \frac{\partial \pi_t(s)}{\partial H_t(a)}$$

Next, we multiply each term of the sum by $\pi_t(x)/\pi_t(x)$:

$$\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \sum_x \pi_t(x) q_*(x) \frac{\partial \pi_t(x)}{\partial H_t(a)} \Big/ \pi_t(x)$$

$$= \mathbb{E}\left[ q_*(A_t) \frac{\partial \pi_t(A_t)}{\partial H_t(a)} \Big/ \pi_t(A_t) \right] = \mathbb{E}\left[ R_t \frac{\partial \pi_t(A_t)}{\partial H_t(a)} \Big/ \pi_t(A_t) \right]$$

This is an expectation, we are summing over all possible values *x* of the random variable $A_t$, then multiplying by the probability of taking those values.

Because, $\mathbb{E}[R_t \mid A_t] = q_*(A_t)$.

$$\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \mathbb{E}\left[ R_t \frac{\partial \pi_t(A_t)}{\partial H_t(a)} \Big/ \pi_t(A_t) \right]$$

# Gradient Bandit Algorithms (Chapter 2) – Derivation

$$\frac{\partial \pi_t(x)}{\partial H_t(a)} = \frac{\partial}{\partial H_t(a)} \pi(x)$$

Let's instantiate $\pi_t(x)$: $\pi_t(x) = \frac{e^{H_t(x)}}{\sum_{y=1}^{k} e^{H_t(y)}}$

$$\frac{\partial \pi_t(x)}{\partial H_t(a)} = \frac{\partial}{\partial H_t(a)} \frac{e^{H_t(x)}}{\sum_{y=1}^{k} e^{H_t(y)}}$$

$$= \frac{\frac{\partial e^{H_t(x)}}{\partial H_t(a)} \sum_{y=1}^{k} e^{H_t(y)} - e^{H_t(x)} \frac{\partial \sum_{y=1}^{k} e^{H_t(y)}}{\partial H_t(a)}}{\left( \sum_{y=1}^{k} e^{H_t(y)} \right)^2}$$

Quotient rule:

$$\frac{\partial}{\partial x}\left[\frac{f(x)}{g(x)}\right] = \frac{\frac{\partial f(x)}{\partial x} g(x) - f(x) \frac{\partial g(x)}{\partial x}}{g(x)^2}$$

$$\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \mathbb{E}\left[ R_t \frac{\partial \pi_t(A_t)}{\partial H_t(a)} \Big/ \pi_t(A_t) \right]$$

# Gradient Bandit Algorithms (Chapter 2) – Derivation

$$\frac{\partial \pi_t(x)}{\partial H_t(a)} = \frac{\frac{\partial e^{H_t(x)}}{\partial H_t(a)} \sum_{y=1}^{k} e^{H_t(y)} - e^{H_t(x)} \frac{\partial \sum_{y=1}^{k} e^{H_t(y)}}{\partial H_t(a)}}{\left( \sum_{y=1}^{k} e^{H_t(y)} \right)^2}$$

$$= \frac{\mathbb{1}_{a=x} e^{H_t(x)} \sum_{y=1}^{k} e^{H_t(y)} - e^{H_t(x)} e^{H_t(a)}}{\left( \sum_{y=1}^{k} e^{H_t(y)} \right)^2}$$

Recall:

$$\frac{\partial e^x}{\partial x} = e^x$$

Thus:

$$\frac{\partial e^{H_t(x)}}{\partial H_t(a)} = \mathbb{1}_{a=x} e^{H_t(x)}$$

that is, when x = y, we have the identity, o.w. we have zero.

$$\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \mathbb{E}\left[ R_t \frac{\partial \pi_t(A_t)}{\partial H_t(a)} \Big/ \pi_t(A_t) \right]$$

# Gradient Bandit Algorithms (Chapter 2) – Derivation

$$\begin{aligned}
\frac{\partial \pi_t(x)}{\partial H_t(a)} &= \frac{\mathbb{1}_{a=x} e^{H_t(x)} \sum_{y=1}^{k} e^{H_t(y)} - e^{H_t(x)} e^{H_t(a)}}{\left( \sum_{y=1}^{k} e^{H_t(y)} \right)^2} \\
&= \frac{\mathbb{1}_{a=x} e^{H_t(x)}}{\sum_{y=1}^{k} e^{H_t(y)}} - \frac{e^{H_t(x)} e^{H_t(a)}}{\left( \sum_{y=1}^{k} e^{H_t(y)} \right)^2} \\
&= \mathbb{1}_{a=x} \pi_t(x) - \pi_t(x)\pi_t(a) \\
&= \pi_t(x)(\mathbb{1}_{a=x} - \pi_t(a))
\end{aligned}$$

# Gradient Bandit Algorithms (Chapter 2) – Derivation

$$\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \mathbb{E}\left[ R_t \frac{\partial \pi_t(A_t)}{\partial H_t(a)} \Big/ \pi_t(A_t) \right]$$

$$\frac{\partial \pi_t(x)}{\partial H_t(a)} = \pi_t(x)(\mathbb{1}_{a=x} - \pi_t(a))$$

$$= \mathbb{E}\left[ \frac{R_t \pi_t(A_t)\left(\mathbb{1}_{a=A_t} - \pi_t(a)\right)}{\pi_t(A_t)} \right]$$

$$= \mathbb{E}\left[ R_t \left(\mathbb{1}_{a=A_t} - \pi_t(a)\right) \right]$$

# Gradient Bandit Algorithms (Chapter 2) – No parameterization

- We change the preferences with stochastic gradient ascent.
- The idea:

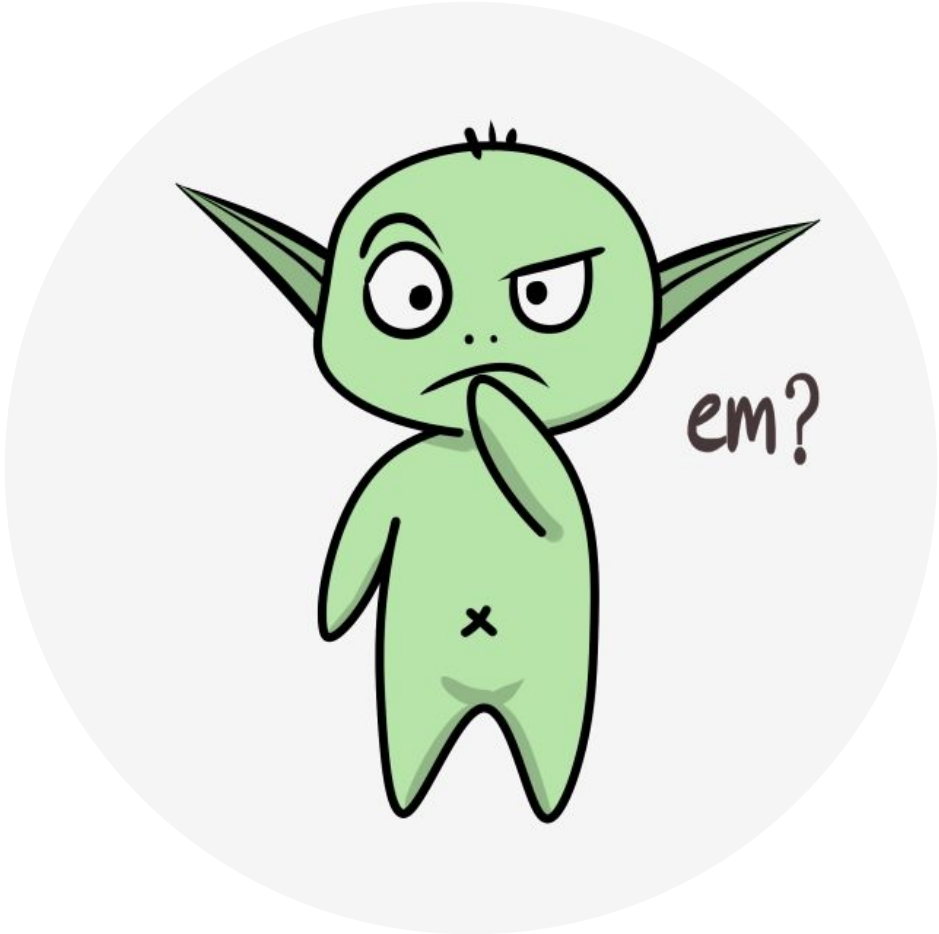
$$H_{t+1}(a) \doteq H_t(a) + \alpha \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} \qquad \text{where } \mathbb{E}[R_t] = \sum_x \pi_t(x) q_*(x)$$

with our derivation, we then have:

$$H_{t+1}(a) = H_t(a) + \alpha \mathbb{E}\Big[R_t \Big(\mathbb{1}_{a=A_t} - \pi_t(a)\Big)\Big]$$

Which means:

$$\begin{aligned} H_{t+1}(A) &\doteq H_t(A_t) + \alpha R_t \Big(1 - \pi_t(A_t)\Big) \quad \text{and} \\ H_{t+1}(a) &\doteq H_t(a) - \alpha R_t \pi_t(a) \quad \text{for all } a \neq A_t. \end{aligned}$$






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