“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 33/35

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.

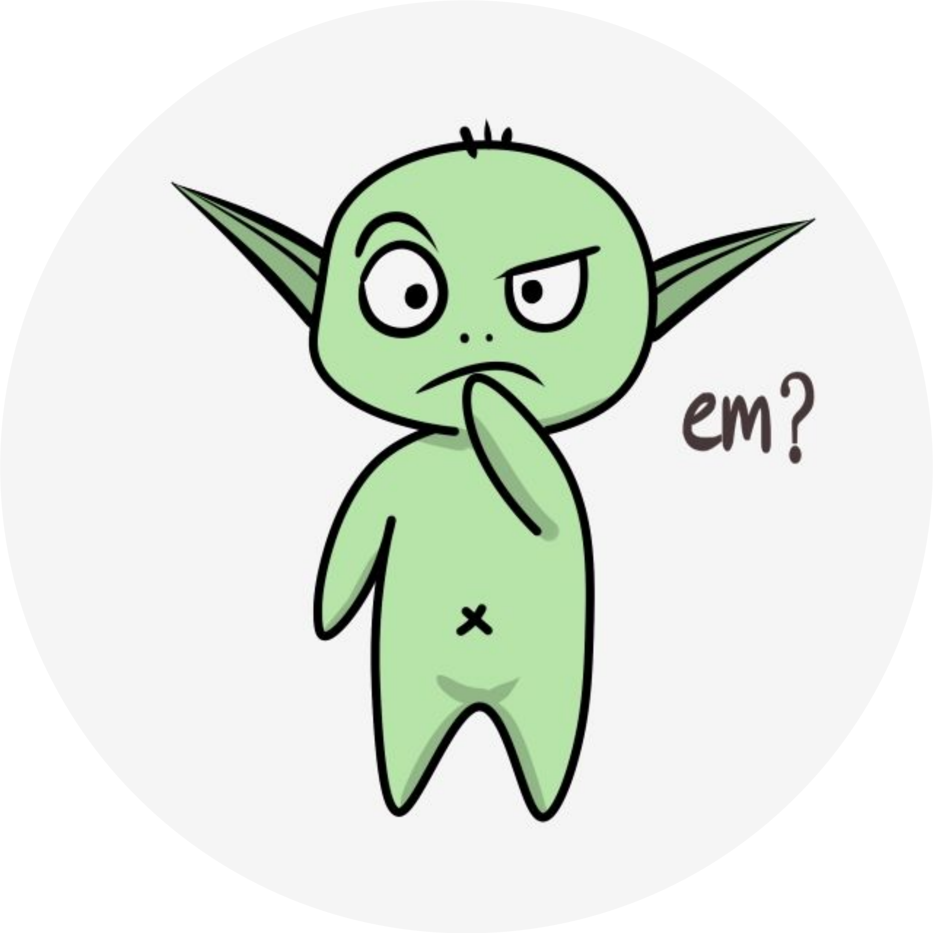
You **need** to **check**, **every time**, if you are in the private session and if you are submitting quizzes and assignments to the private section.

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:
  - This week, my office hours will be **shorter** on Thursday: 2pm to 3pm
  - Next week, it will be **shifted** to 12:30pm to 2:30pm on Thursday
- The **last** quiz and programming assignment are due today.
- 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

Last Class: Gradient Bandit Algorithms (Chapter 2)

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

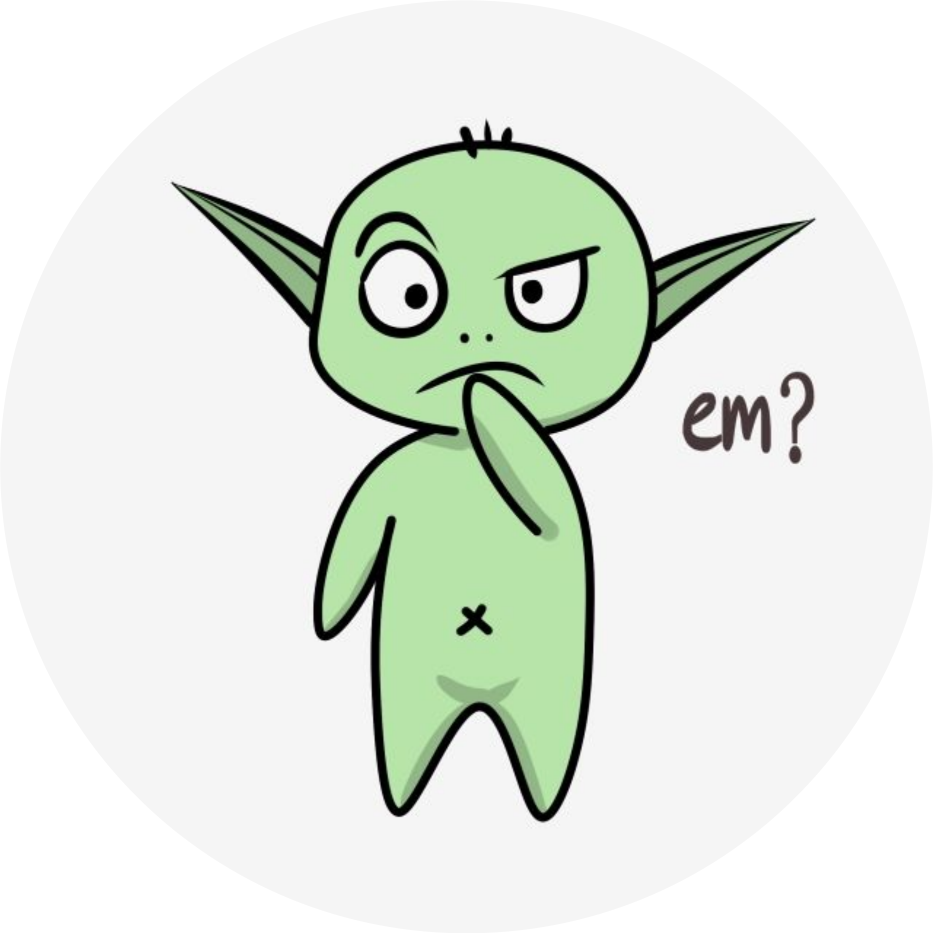
$$H_{t+1}(a) \doteq H_t(a) + \alpha \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} \qquad \text{where } \mathbb{E}[R_t] = \sum_x \pi_t(x) q_*(x)$$

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

Gradient Bandit Algorithms (Chapter 2) – Baseline

Let’s look at the exact performance gradient:

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

Instead, we could do:

Baseline: Any scalar that does not depend on *x*.

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

We can do this because the gradient sums to zero over all the actions, $\sum_x \frac{\partial \pi_t(x)}{\partial H_t(a)} = 0$. As $H_t(a)$ is changed, some actions’ prob. go up and some go down, but the sum of the changes must be zero because the sum of the prob. Is always 1.

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

Average of the rewards up t

but not including time *t*.

$$\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \sum_x \pi_t(x)\Big(q_*(x) - B_t\Big)\frac{\partial \pi_t(x)}{\partial H_t(a)}\Big/\pi_t(x) = \mathbb{E}\left[\Big(q_*(x) - B_t\Big)\frac{\partial \pi_t(A_t)}{\partial H_t(a)}\Big/\pi_t(A_t)\right] = \mathbb{E}\left[\Big(R_t - \bar{R}_t\Big)\frac{\partial \pi_t(A_t)}{\partial H_t(a)}\Big/\pi_t(A_t)\right]$$

Gradient Bandit Algorithms (Chapter 2) – No parameterization

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

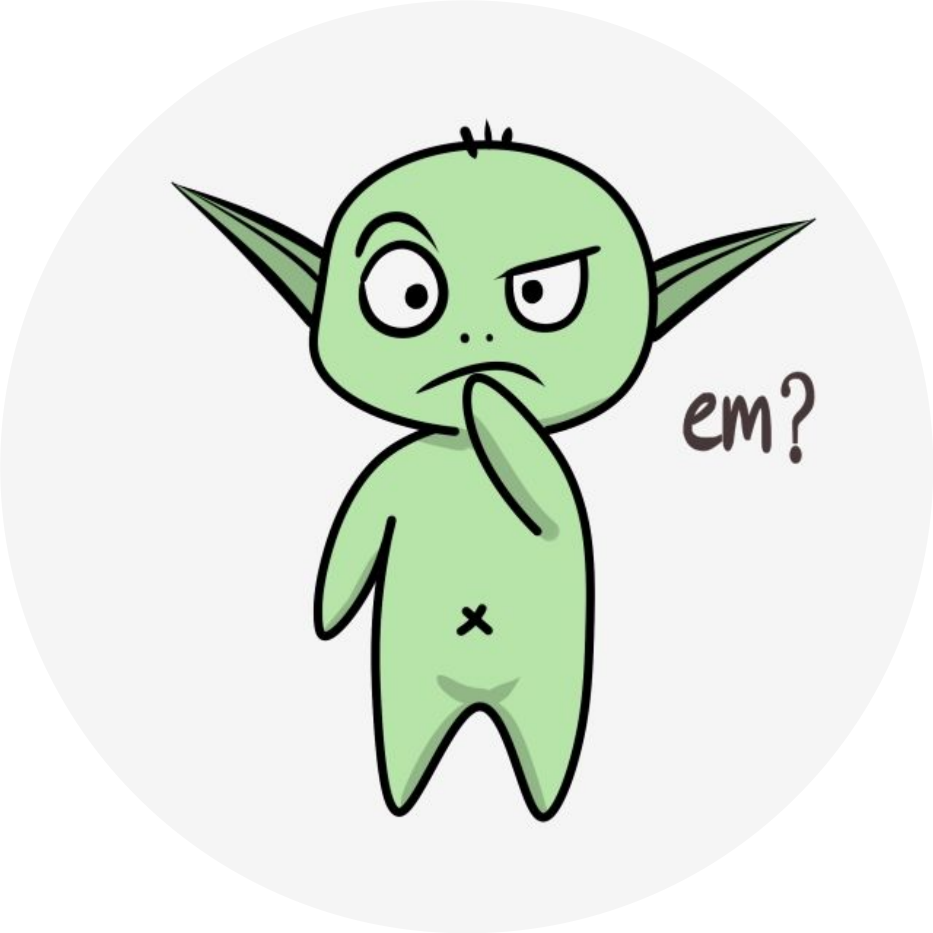
$$H_{t+1}(a) \doteq H_t(a) + \alpha \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} \qquad \text{where } \mathbb{E}[R_t] = \sum_x \pi_t(x) q_*(x)$$

with our derivation, we then have:

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

Which means:

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



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

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?
  - We parameterize a policy and we learn the values of those parameters ¯_(ツ)_/¯
  - We can still use a value function to learn the policy params, but it isn’t required for action selection.

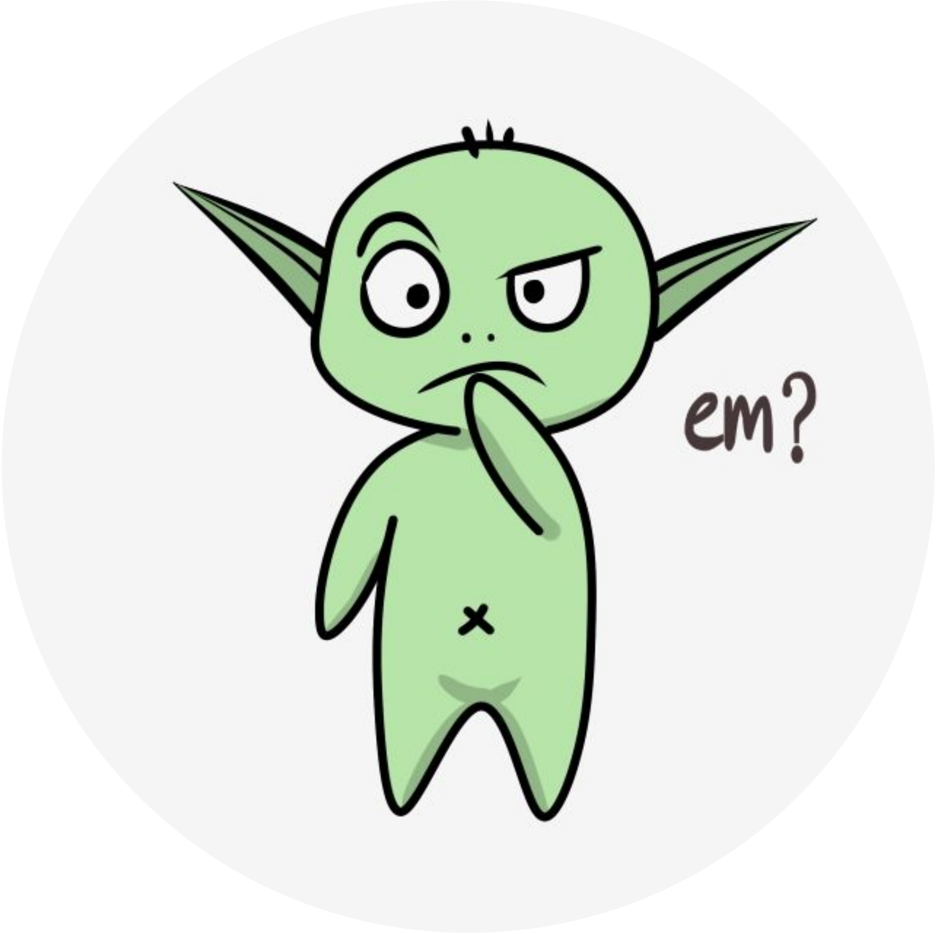
More Specifically

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

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

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

- Methods that learn approximations to both policy and value functions are often called actor-critic methods.
  - Actor: learned policy
  - Critic: learned value function



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

Scaling up and Generalizing Policy Gradient Methods

- In the Bandits problem, we used the following softmax:

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

- More generally, we need to introduce states and to parameterize H_t:

$$\pi(a|s, \boldsymbol{\theta}) \doteq \frac{e^{h(s,a,\boldsymbol{\theta})}}{\sum_b e^{h(s,b,\boldsymbol{\theta})}}$$

This could be "anything", such as a neural network or linear in features:

$$e^{h(s,a,\boldsymbol{\theta})} = e^{\boldsymbol{\theta}^\top \mathbf{x}(s,a)}$$

Some Advantages of Policy Approximation

- The softmax in action preferences can approach a deterministic policy, whereas ε-greedy, for example, will always have a probability ε of acting suboptimally.
  - These decisions are made per “state” with policy gradient methods, not globally.

Some Advantages of Policy Approximation

- The softmax in action preferences can approach a deterministic policy, whereas ε-greedy, for example, will always have a probability ε of acting suboptimally.
  - These decisions are made per “state” with policy gradient methods, not globally.

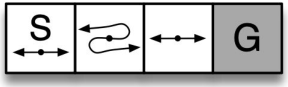
- It is not the softmax itself that gives us that, but the use of action preferences. A softmax over state-action values would never converge to a deterministic policy.
  - Action preferences are driven to produce the optimal *stochastic* policy.

Some Advantages of Policy Approximation

- The softmax in action preferences can approach a deterministic policy, whereas ε-greedy, for example, will always have a probability ε of acting suboptimally.
  - These decisions are made per “state” with policy gradient methods, not globally.

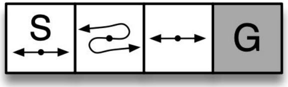
- It is not the softmax itself that gives us that, but the use of action preferences. A softmax over state-action values would never converge to a deterministic policy.z
  - Action preferences are driven to produce the optimal *stochastic* policy.

- We can select actions with arbitrary probabilities. Sometimes the best policy is a stochastic one.
  - See Example 13.1 of the textbook

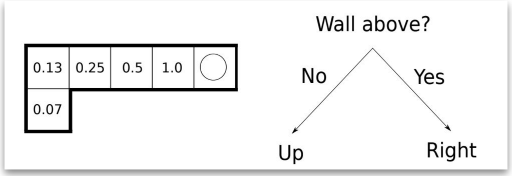


Some Advantages of Policy Approximation

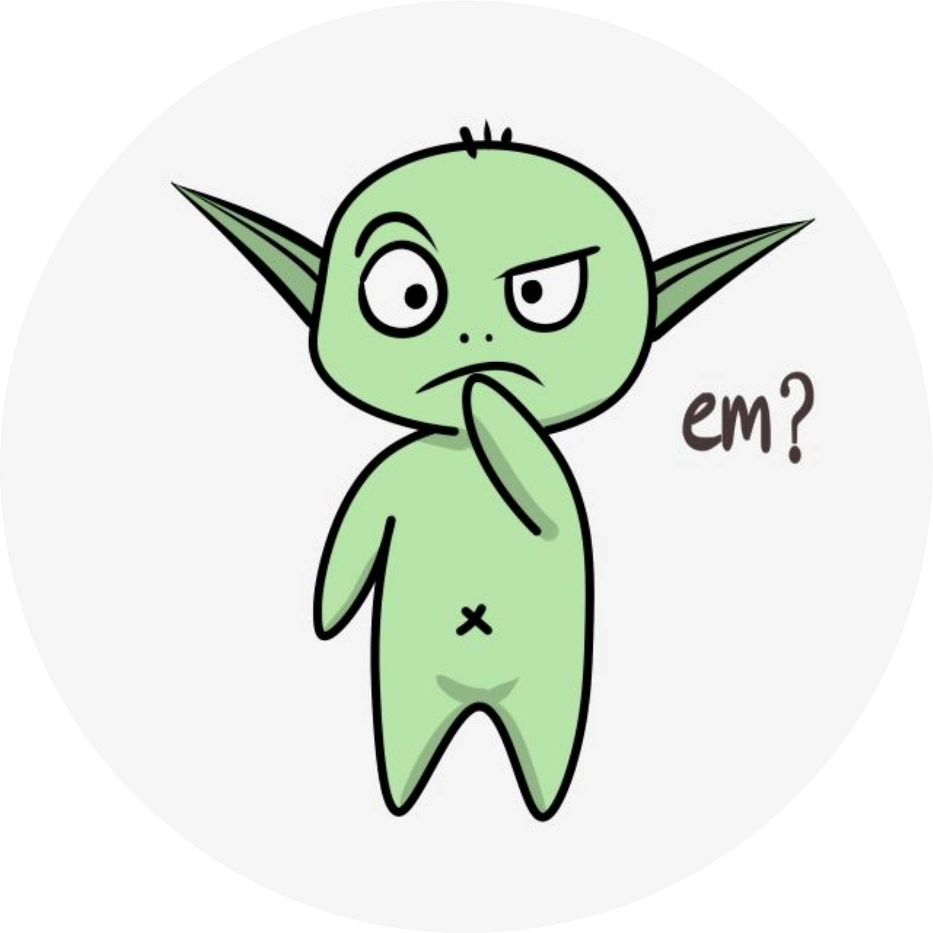
- The softmax in action preferences can approach a deterministic policy, whereas ε-greedy, for example, will always have a probability ε of acting suboptimally.
  - These decisions are made per "state" with policy gradient methods, not globally.
- It is not the softmax itself that gives us that, but the use of action preferences. A softmax over state-action values would never converge to a deterministic policy.z
  - Action preferences are driven to produce the optimal *stochastic* policy.
- We can select actions with arbitrary probabilities. Sometimes the best policy is a stochastic one.
  - See Example 13.1 of the textbook



- A policy might be a easier to approximate than the VF. And it is the thing you ultimately care about ¯_(ツ)_/¯



[Das Gupta, Talvitie, Bowling; AAAI 2015]



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

The Policy Gradient Theorem

- We have stronger convergence guarantees for policy gradient methods, in part because the policy changes more smoothly than, say, ε-greedy policies.
- Let's do gradient ascent, in the full RL problem. To do that we need to define J(**θ**):

$$J(\boldsymbol{\theta}) \doteq v_{\pi_{\boldsymbol{\theta}}}(s_0)$$

- It seems tricky though. "The problem is that performance depends on both the action selections and the distribution of states in which those selections are made, and that both of these are affected by the policy parameter."
    - The effect of a change in the policy on the state distribution is typically unknown.

The Policy Gradient Theorem

- We have stronger convergence guarantees for policy gradient methods, in part because the policy changes more smoothly than, say, ε-greedy policies.

- Let's do gradient ascent, in the full RL problem. To do that we need to define J(**θ**):

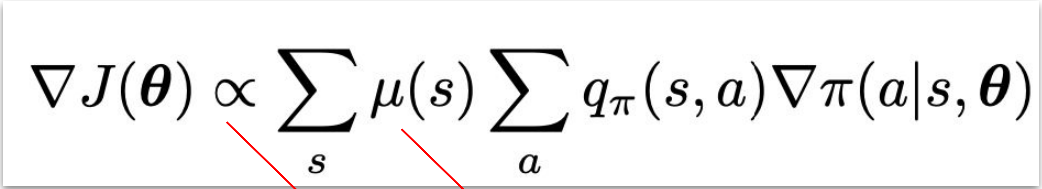
$$J(\boldsymbol{\theta}) \doteq v_{\pi_{\boldsymbol{\theta}}}(s_0)$$

- It seems tricky though. "The problem is that performance depends on both the action selections and the distribution of states in which those selections are made, and that both of these are affected by the policy parameter."
  - The effect of a change in the policy on the state distribution is typically unknown.

How can we estimate the performance gradient w.r.t the policy parameter when the gradient depends on the unknown effect of policy changes on the state distribution?

The Policy Gradient Theorem

- The Policy Gradient Theorem [Marbach and Tsitsiklis, 1998, 2001; Sutton et al. 2000] provides an analytic expression for the gradient of performance w.r.t. the policy parameter that does not involve the derivative of the state distribution.



$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$

Proportional to

On-policy distribution

Interlude: The On-Policy Distribution (Page 199)

[In episodic tasks, without discounting.]

- In an episodic task, the on-policy distribution depends on the initial state distribut.
- Let *h(s)* denote the probability that an episode begins in each state *s.*
- Let *η(s)* denote the # of time steps spent, on avg, in state *s* in a single episode.

  Time is spent in a state *s* if episodes start in *s*, or if transitions are made into *s* from a preceding state $\bar{s}$ in which time is spent.

$$\eta(s) \quad = \quad h(s) + \sum_{\bar{s}} \eta(\bar{s}) \sum_a \pi(a|\bar{s}) p(s|\bar{s}, a), \quad \text{for all } s \in \mathcal{S}.$$

Interlude: The On-Policy Distribution (Page 199)

[In episodic tasks, without discounting.]

- In an episodic task, the on-policy distribution depends on the initial state distribut.
- Let *h(s)* denote the probability that an episode begins in each state *s.*
- Let *η(s)* denote the # of time steps spent, on avg, in state *s* in a single episode.

  Time is spent in a state *s* if episodes start in *s*, or if transitions are made into *s* from a preceding state $\bar{s}$ in which time is spent.

$$\eta(s) \quad = \quad h(s) + \sum_{\bar{s}} \eta(\bar{s}) \sum_{a} \pi(a|\bar{s}) p(s|\bar{s}, a), \quad \text{for all } s \in \mathcal{S}.$$

- Solving this system of equations we obtain the on-policy distribution, which is the fraction of time spent in each state normalized to sum to one:

$$\mu(s) \quad = \quad \frac{\eta(s)}{\sum_{s'} \eta(s')}, \quad \text{for all } s \in \mathcal{S}.$$

The Policy Gradient Theorem – Proof

[For simplicity, we leave it implicit that π is a function of **θ**, and all gradients are also implicitly with respect to **θ**.]

$$
\begin{aligned}
\nabla v_\pi(s) &= \nabla\left[\sum_a \pi(a|s) q_\pi(s,a)\right], \quad \text{for all } s \in \mathcal{S} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s)\nabla q_\pi(s,a)\Big] \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s)\nabla \sum_{s',r} p(s',r|s,a)\big(r + v_\pi(s')\big)\Big] \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a)\nabla v_\pi(s')\Big] \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \\
&\qquad \sum_{a'} \big[\nabla\pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a')\nabla v_\pi(s'')\big]\Big]
\end{aligned}
$$

Reminder
Product rule

$$\nabla(fg) = f\nabla g + g\nabla f$$

The Policy Gradient Theorem – Proof

[For simplicity, we leave it implicit that π is a function of **θ**, and all gradients are also implicitly with respect to **θ**.]

k = 0 k = 1

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

If we keep unrolling, considering all time steps:

$$= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \rightarrow x, k, \pi) \sum_a \nabla \pi(a|x) q_\pi(x,a)$$

Probability of transitioning from state s to state x in k steps under policy π.

The Policy Gradient Theorem – Proof

[For simplicity, we leave it implicit that π is a function of **θ**, and all gradients are also implicitly with respect to **θ**.]

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \nabla v_\pi(s) \\
&= \sum_s \underbrace{\left(\sum_{k=0}^{\infty} \Pr(s_0 \rightarrow s, k, \pi) \right)}_{\text{\# of time steps spent on}} \sum_a \nabla \pi(a|x) q_\pi(x, a) \\
&= \sum_s \eta(s) \sum_a \nabla \pi(a|x) q_\pi(x, a)
\end{aligned}
$$

Next, we multiply everything by $\sum_{s'} \eta(s') \Big/ \sum_{s'} \eta(s')$

$$
= \sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_a \nabla \pi(a|x) q_\pi(x, a)
$$

The Policy Gradient Theorem – Proof

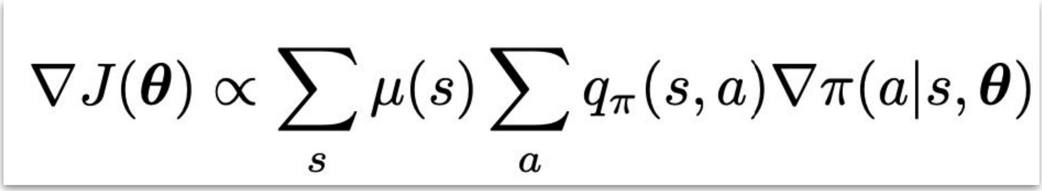
[For simplicity, we leave it implicit that π is a function of **θ**, and all gradients are also implicitly with respect to **θ**.]

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \sum_{s'} \eta(s') \underline{\sum_{s} \frac{\eta(s)}{\sum_{s'} \eta(s')}} \sum_{a} \nabla \pi(a|x) q_\pi(x,a) \\
&= \sum_{s'} \eta(s') \sum_{s} \mu(s) \sum_{a} \nabla \pi(a|x) q_\pi(x,a) \\
&\propto \sum_{s} \mu(s) \sum_{a} \nabla \pi(a|x) q_\pi(x,a)
\end{aligned}
$$

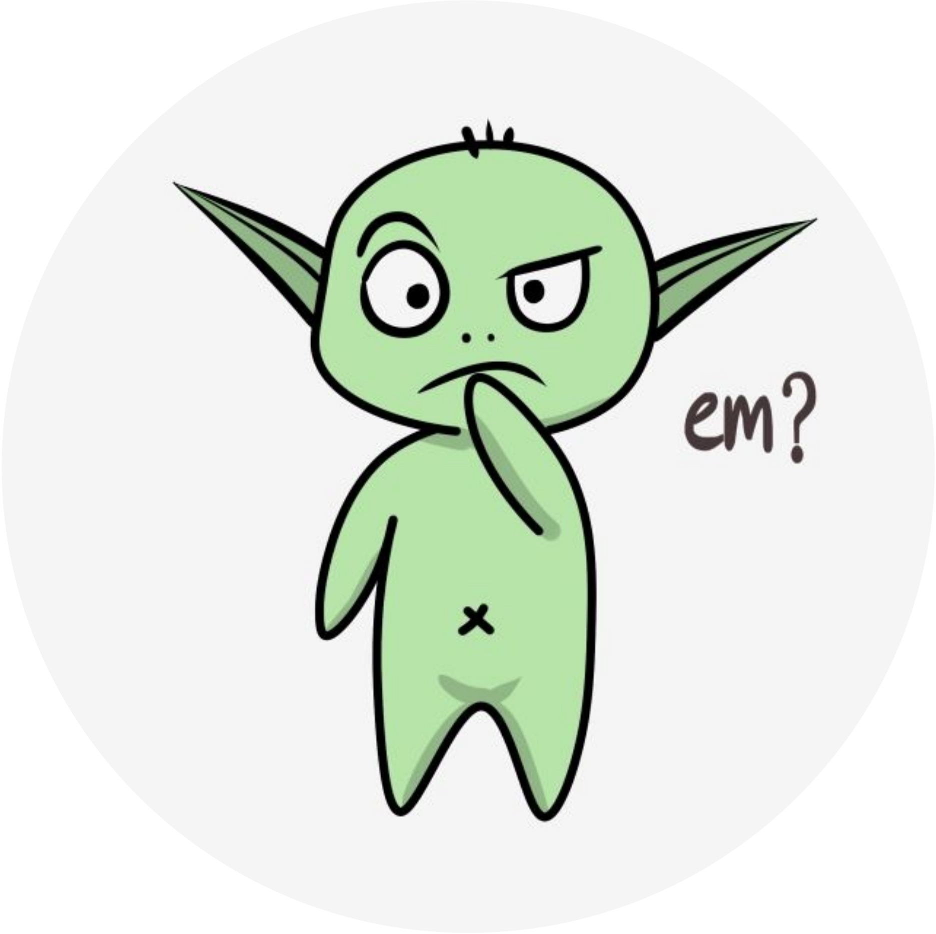
Q.E.D.

The Policy Gradient Theorem

- The Policy Gradient Theorem [Marbach and Tsitsiklis, 1998, 2001; Sutton et al. 2000] provides an analytic expression for the gradient of performance w.r.t. the policy parameter that does not involve the derivative of the state distribution.



$$\nabla J(\boldsymbol{\theta}) \propto \sum_{s} \mu(s) \sum_{a} q_{\pi}(s, a) \nabla \pi(a|s, \boldsymbol{\theta})$$



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