“And always, he fought the temptation to choose a clear, safe course, warning “That path leads ever down into stagnation.””

Frank Herbert, *Dune*

CMPUT 655
Introduction to Sequential-Decision Making

Marlos C. Machado

Plan

- Motivation
- *Non-comprehensive* overview of Intro to Sequential-Decision Making in Coursera (Bandits, Chapter 2 of the textbook)
- Proof of UCB1's regret bound
- Gradient Bandit Algorithms

Please, interrupt me at any time!

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

Let’s play a game!

https://m.media-amazon.com/images/I/91q+3X0sUZL._AC_SL1500_.jp

Bandits

Arm 1	Arm 2	Arm 3

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 said in the videos, but more generally about sequential decision-making.

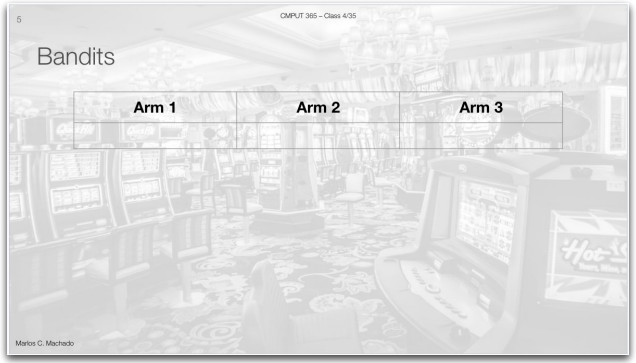
- Reinforcement learning potentially allows for continual learning but in practice, quite often we deploy our systems.

Why study bandits?

- Bandits are the simplest possible reinforcement learning problem.
  - Actions have no delayed consequences.

- Bandits are deployed in so many places! [Source: Csaba's slides]
  - Recommender systems (Microsoft paper):
    - News,
    - Videos,
    - …
  - Targeted COVID-19 border testing (Deployed in Greece, paper).
  - Adapting audits (Being deployed at IRS in the USA, paper).
  - Customer support bots (Microsoft paper).
  - … and more.

Why study bandits?

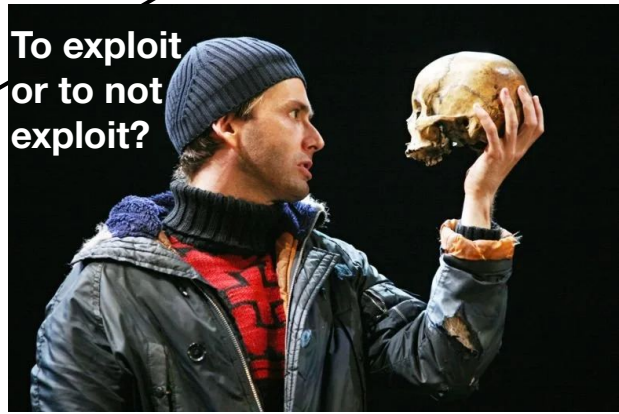


We don't really know q*, so we use an estimate of it, Q_t

$$q^*(a) \doteq \mathbb{E}[R_t \mid A_t = a]$$

$$A_t \doteq \text{argmax}_a \, Q_t(a)$$

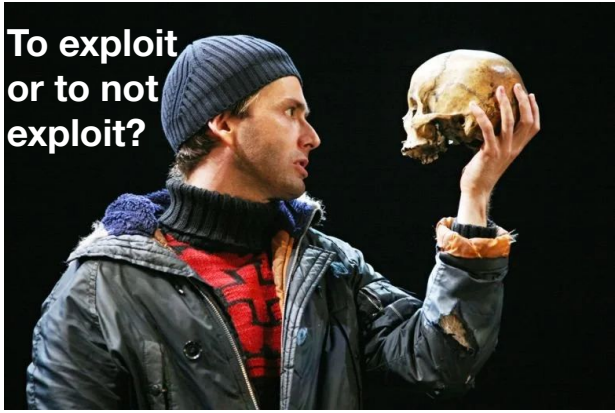
Greedy action



https://shakespeareanstudent.files.wordpress.com/2019/10/fb89ff1a-6f63-11e6-acba-85f5c900fc1a1735507721165554264.

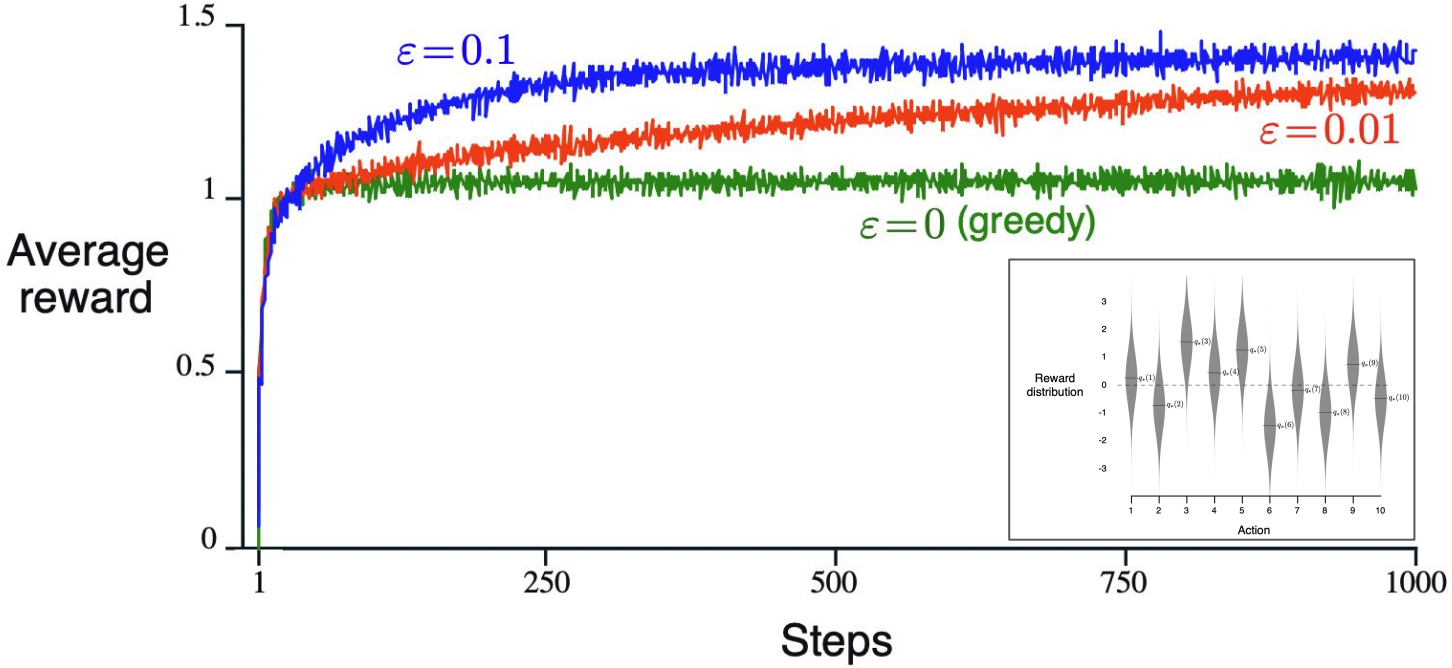
Exploration

- Exploration is the opposite of exploitation.
- It is a whole, very active area of research, despite the textbook not focusing on it.
- How can we explore?
  - Randomly ($\in$-greedy)
  - Optimism in the face of uncertainty
  - Uncertainty
  - Novelty / Boredom / Surprise
  - Temporally-extended exploration
  - …



https://shakespeareanstudent.files.wordpress.com/2019/10/fb89ff1a-6f63-11e6-acba-85f5c900fc1a1735507721165554264.

Exploration matters



Incremental updates to estimate q_*

$$Q_{n+1} = \frac{1}{n}\sum_{i=1}^{n} R_i$$

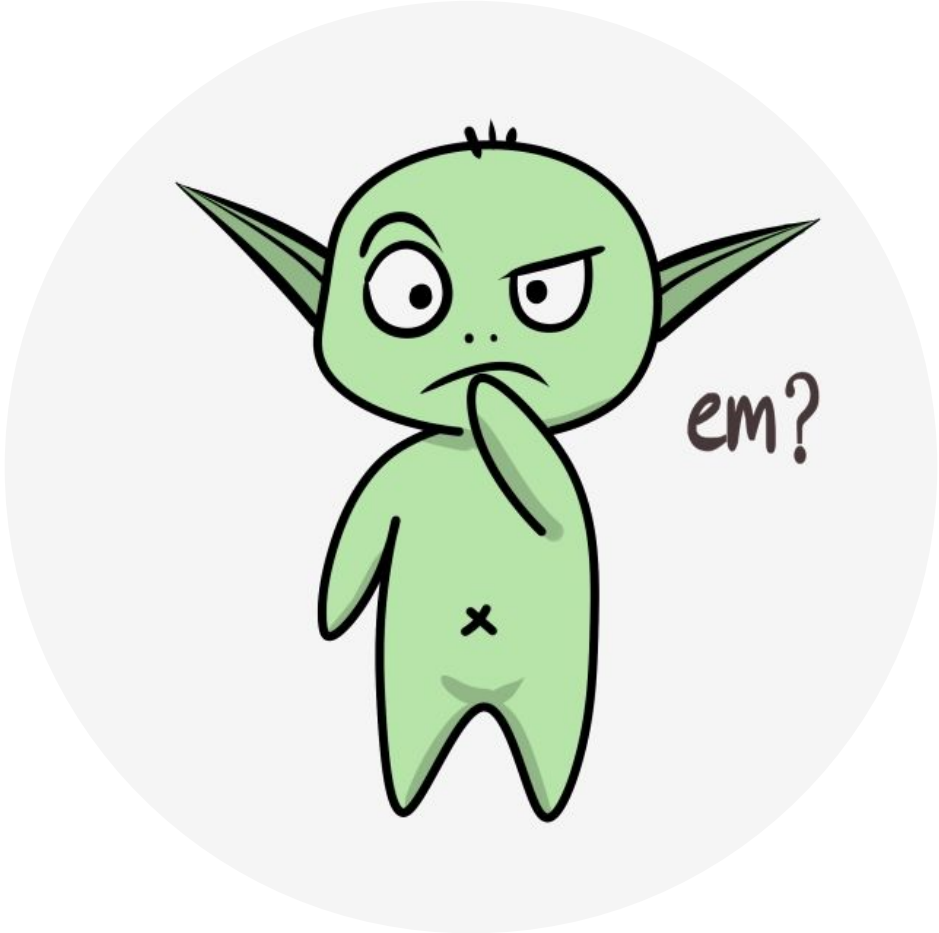
Incremental updates to estimate q_*

$$
\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\Big(R_n + (n-1)Q_n\Big) \\
&= \frac{1}{n}\Big(R_n + nQ_n - Q_n\Big) \\
&= Q_n + \frac{1}{n}\Big[R_n - Q_n\Big]
\end{aligned}
$$

Incremental updates to estimate q_*

$$
\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\left(R_n + (n-1)Q_n\right) \\
&= \frac{1}{n}\left(R_n + nQ_n - Q_n\right) \\
&= Q_n + \frac{1}{n}\left[R_n - Q_n\right]
\end{aligned}
$$

NewEstimate ← OldEstimate + StepSize [Target - OldEstimate]



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

Update rule

NewEstimate ← OldEstimate + StepSize [Target - OldEstimate]

$$Q_{n+1} \doteq Q_n + \alpha [R_n - Q_n]$$

A bigger step-size means bigger steps (updates).

A constant step-size gives more weight to recent rewards.

How you initialize Q_n really matters.

The principle of **optimism in the face of uncertainty** really leverages that.

This is the direction you need to move to get closer to the solution.

A note on step-sizes

A well-known result in stochastic approximation theory gives us the conditions required to assure convergence with probability 1:

$$\sum_{n=1}^{\infty} \alpha_n(a) = \infty \quad \text{and} \quad \sum_{n=1}^{\infty} \alpha_n^2(a) < \infty$$

Cannot be too small.
E.g.: $\alpha_n = 1/n^2$

Cannot be too big.
E.g.: $\alpha_n = 1$

A constant step-size is biased

$$Q_{n+1} = Q_n + \alpha\Big[R_n - Q_n\Big]$$

A constant step-size is biased

$$
\begin{aligned}
Q_{n+1} &= Q_n + \alpha\Big[R_n - Q_n\Big] \\
&= \alpha R_n + (1-\alpha)Q_n \\
&= \alpha R_n + (1-\alpha)\left[\alpha R_{n-1} + (1-\alpha)Q_{n-1}\right] \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha R_{n-2} + \\
&\qquad \cdots + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n Q_1 \\
&= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i .
\end{aligned}
$$

Q_1 is always there, forever, impacting the final estimate.

Exercise 2.7 of the textbook

Exercise 2.7: Unbiased Constant-Step-Size Trick In most of this chapter we have used sample averages to estimate action values because sample averages do not produce the initial bias that constant step sizes do (see the analysis leading to (2.6)). However, sample averages are not a completely satisfactory solution because they may perform poorly on nonstationary problems. Is it possible to avoid the bias of constant step sizes while retaining their advantages on nonstationary problems? One way is to use a step size of

$$\beta_n \doteq \alpha/\bar{o}_n, \tag{2.8}$$

to process the nth reward for a particular action, where $\alpha > 0$ is a conventional constant step size, and $\bar{o}_n$ is a trace of one that starts at 0:

$$\bar{o}_n \doteq \bar{o}_{n-1} + \alpha(1 - \bar{o}_{n-1}), \quad \text{for } n > 0, \quad \text{with } \bar{o}_0 \doteq 0. \tag{2.9}$$

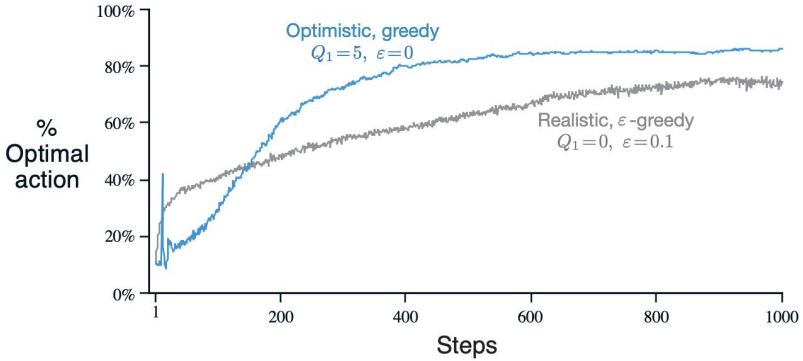
Carry out an analysis like that in (2.6) to show that Q_n is an exponential recency-weighted average *without initial bias*. □

Solution – Exercise 2.7 of the textbook

Optimism in the face of uncertainty

$$Q_{n+1} = Q_n + \alpha\left[R_n - Q_n\right]$$

Idea: Initialize Q_0 to an overestimation of its true value (optimistically).

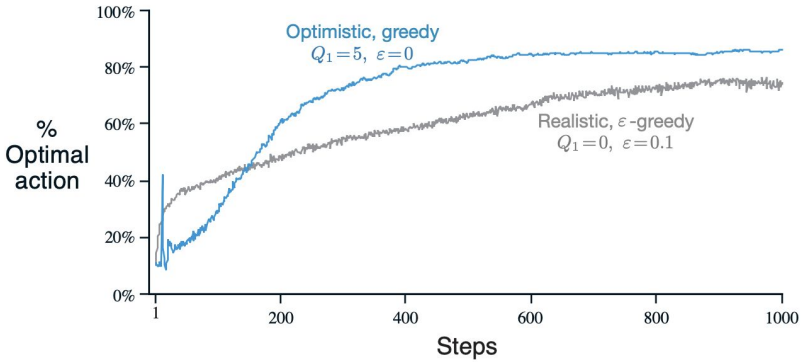


Optimism in the face of uncertainty

$$Q_{n+1} = Q_n + \alpha\left[R_n - Q_n\right]$$

Idea: Initialize Q_0 to an overestimation of its true value (optimistically).

- You either maximize reward or you learn from it.
- The value you initialize Q_0 can be seen as a hyperparameter and it matters.
- There are equivalent transformations in the reward signal to get the same effect.
- For bandits, UCB uses an upper confidence bound that with high probability is an overestimate of the unknown value.



Upper-Confidence-Bound Action Selection

$$A_t \doteq \arg\max_a \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

Theorem 1. *For all $K > 1$, if policy* UCB1 *is run on K machines having arbitrary reward distributions $P_1, \ldots, P_K$ with support in $[0, 1]$, then its expected regret after any number n of plays is at most*

$$\left[8 \sum_{i:\mu_i < \mu^*} \left(\frac{\ln n}{\Delta_i} \right) \right] + \left(1 + \frac{\pi^2}{3} \right) \left(\sum_{j=1}^{K} \Delta_j \right)$$

where $\mu_1, \ldots, \mu_K$ are the expected values of $P_1, \ldots, P_K$.

Auer, Cesa-Bianchi, and Fischer (2002), *Machine Learning*.

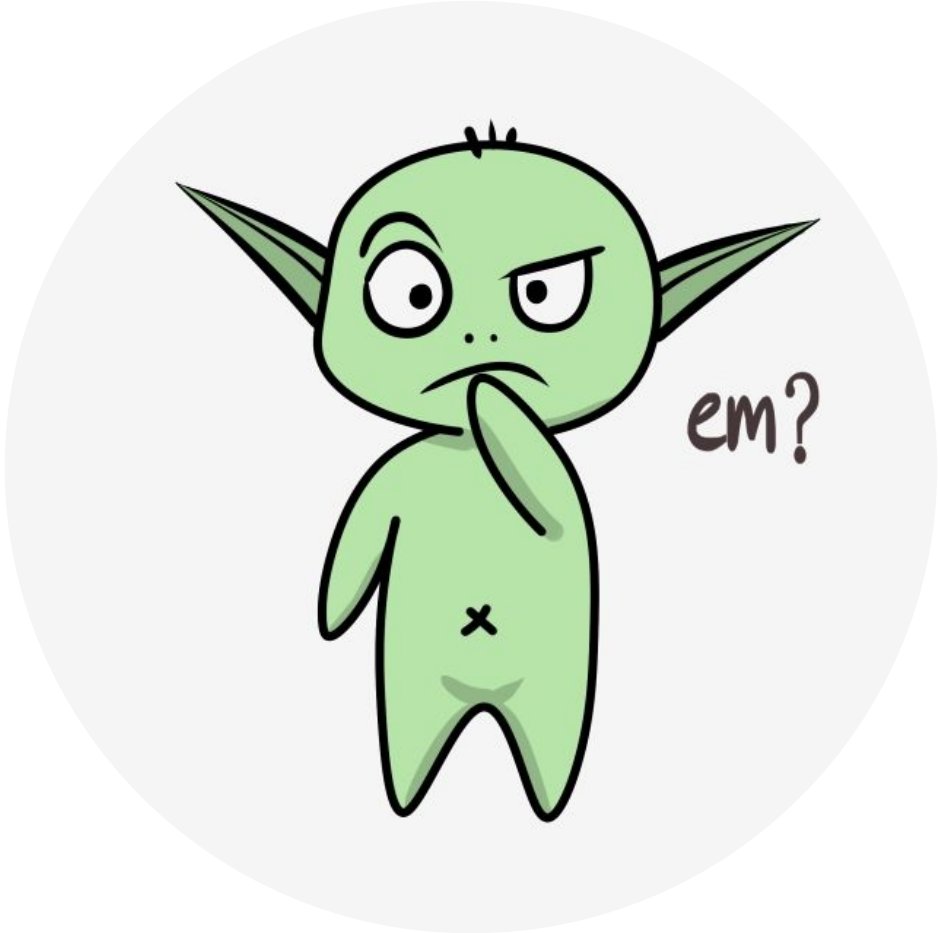
Proof of UCB1's regret bound

Deterministic policy: UCB1.
Initialization: Play each machine once.
Loop:

- Play machine j that maximizes $\bar{x}_j + \sqrt{\frac{2 \ln n}{n_j}}$, where $\bar{x}_j$ is the average reward obtained from machine j, n_j is the number of times machine j has been played so far, and n is the overall number of plays done so far.

Proof of UCB1's regret bound

Proof of UCB1's regret bound



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

Gradient Bandit Algorithms

- Instead of learning action values and using those estimates to select actions, we can instead learn a numerical preference, $H_t(a)$, for each action a.
- Only the relative preference of one action over another is important.

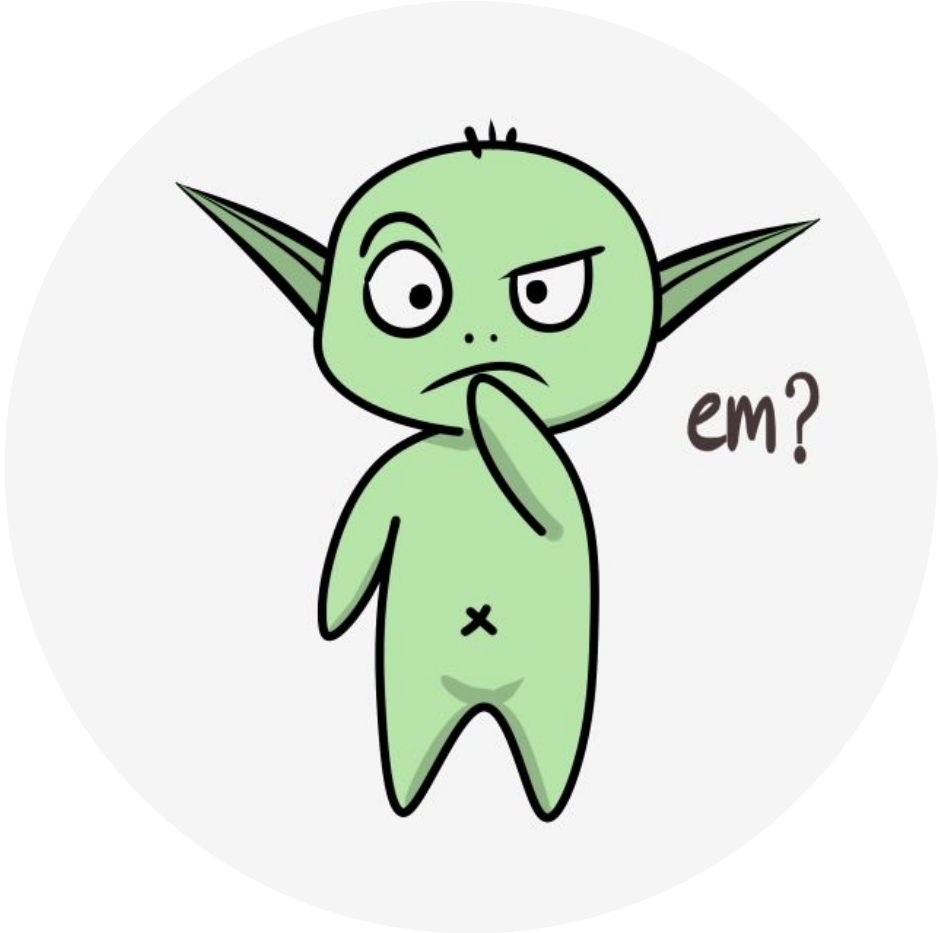
$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

Policy: prob. of taking action a at time t.

- There is a natural learning algorithm for this setting based on the idea of stochastic gradient ascent:

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), \quad \text{and}$$
$$H_{t+1}(a) \doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), \quad \text{for all } a \neq A_t$$

Gradient Bandit Algorithms – Derivation of update rule



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

Contextual bandits (Associative search)

In every round t, the learner first observes context $\mathbf{c}_t(a) \in \mathscr{C}$ *for every arm* $a \in \mathscr{A}$*, then chooses an arm* a_t*, and finally observes a reward* $r_t(a_t)$*.*

Contextual bandits (Associative search)

In every round t, the learner first observes context $\mathbf{c}_t(a) \in \mathscr{C}$ *for every arm* $a \in \mathscr{A}$*, then chooses an arm* a_t*, and finally observes a reward* $r_t(a_t)$*.*

- One needs to associate different actions with different *situations*.
- You need to learn a *policy,* which is a function that maps situations to actions.
- Most real-world problems modeled as bandits problems are modeled as contextual bandits problems.
- Example: A recommendation system, which is obviously conditioned on the user to which the system is making recommendations to.

Contextual bandits (Associative search)

In every round t, the learner first observes context $\mathbf{c}_t(a) \in \mathscr{C}$ *for every arm* $a \in \mathscr{A}$*, then chooses an arm* a_t*, and finally observes a reward* $r_t(a_t)$*.*

If we want to use linear function approximation, for example (roughly):

$\pi_{\mathbf{w}}(\mathbf{c}(\cdot)) = \Delta(\mathscr{A})$ — policy

$\mathbf{c}(a_1) = [1, 0, 3.4, 20, 1\ ]^\top$

$\mathbf{c}(a_2) = [0, 1, 4.2, 13, 0\ ]^\top$

$\mathbf{c}(a_3) = [1, 1, 2.7, 66, 0\ ]^\top$

Contextual bandits (Associative search)

In every round t, the learner first observes context $\mathbf{c}_t(a) \in \mathscr{C}$ *for every arm* $a \in \mathscr{A}$*, then chooses an arm* a_t*, and finally observes a reward* $r_t(a_t)$*.*

If we want to use linear function approximation, for example (roughly):

$\pi_{\mathbf{w}}(\mathbf{c}(\cdot)) = \Delta(\mathscr{A})$

$\mathbf{c}(a_1) = [1, 0, 3.4, 20, 1\ ]^\top$

$\mathbf{c}(a_2) = [0, 1, 4.2, 13, 0\ ]^\top$

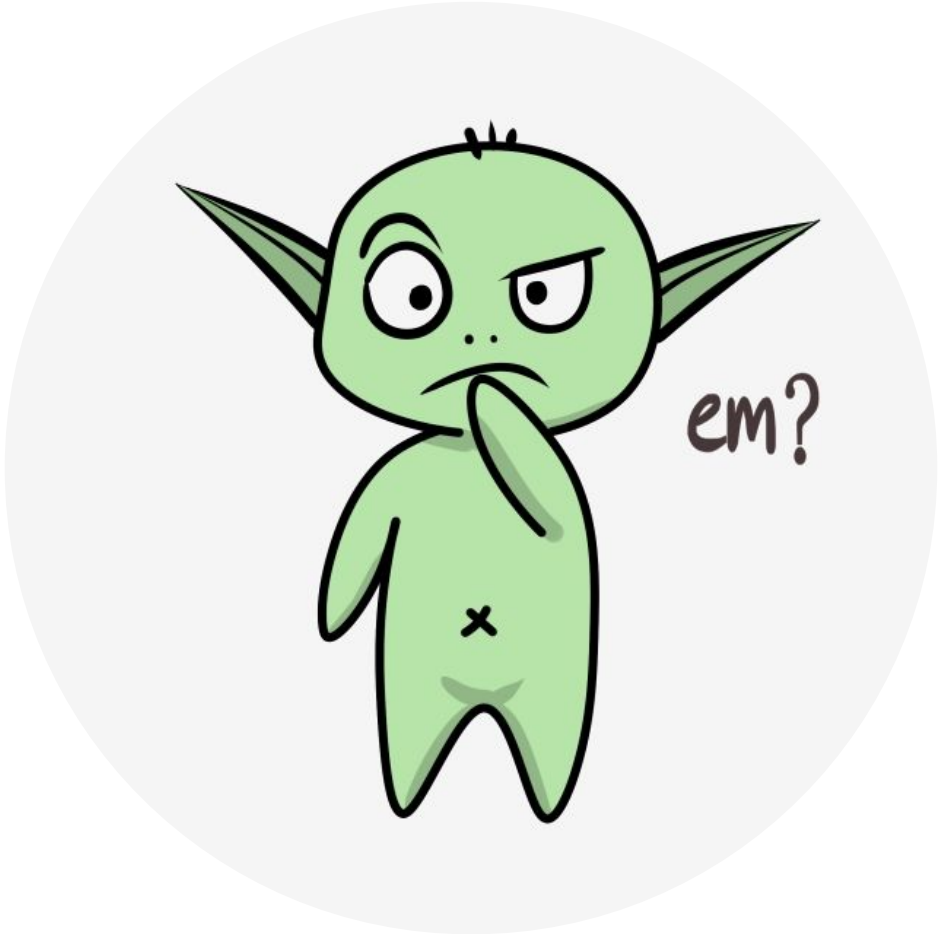
$\mathbf{c}(a_3) = [1, 1, 2.7, 66, 0\ ]^\top$

$\mathbf{w} = [0.3, 0.5, 0.1, 0.01, 0.4]^\top$

$q(a_1) = \mathbf{w}^\top \mathbf{c}(a_1) = 1.24$

$q(a_2) = \mathbf{w}^\top \mathbf{c}(a_2) = 0.55$

$q(a_3) = \mathbf{w}^\top \mathbf{c}(a_3) = 1.23$



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

Next class

Reminder: Programming assignment for Coursera's Fundamentals of RL: Sequential decision-making is due today at midnight.

- What **I** plan to do: Wrap up Fundamentals of RL: An introduction to sequential decision-making (Bandits)
  - Go over some of your questions from Slack and eClass.
  - Time permitting, we'll work on some exercises in the classroom.

- What I recommend **YOU** to do for next class:
  - If you haven't yet:
    - Start Fundamentals of RL: Markov Decision Processes (Week 2 in Coursera)
    - Practice quiz is due on Monday, but then that's it for the week 😄