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

Frank Herbert, *Dune*

CMPUT 365
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)

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.

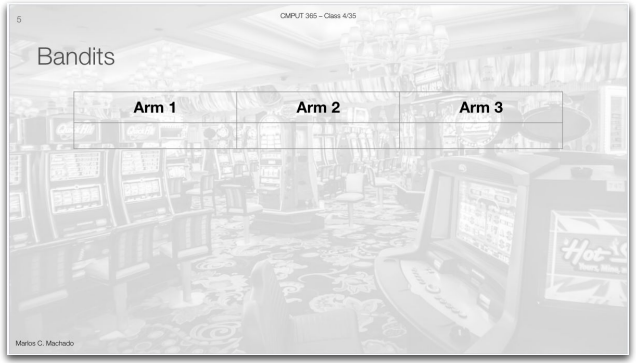
- Reinforcement learning potentially allows for continual learning but in practice, quite often we deploy our systems.

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?

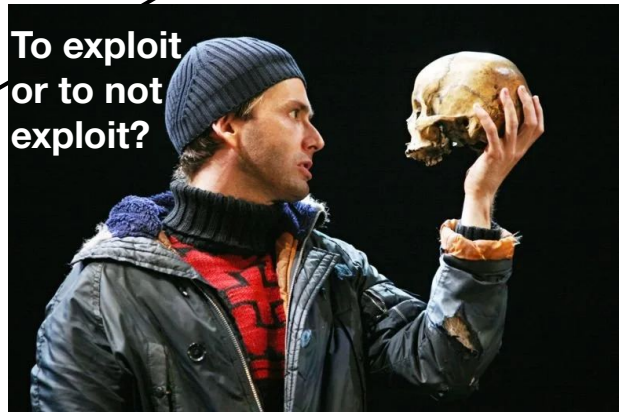


We don't really know q*, so we use an estimate of it, Q_t

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

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

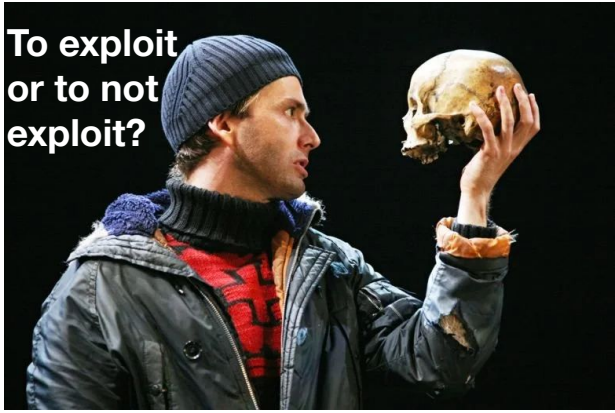
Greedy action



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

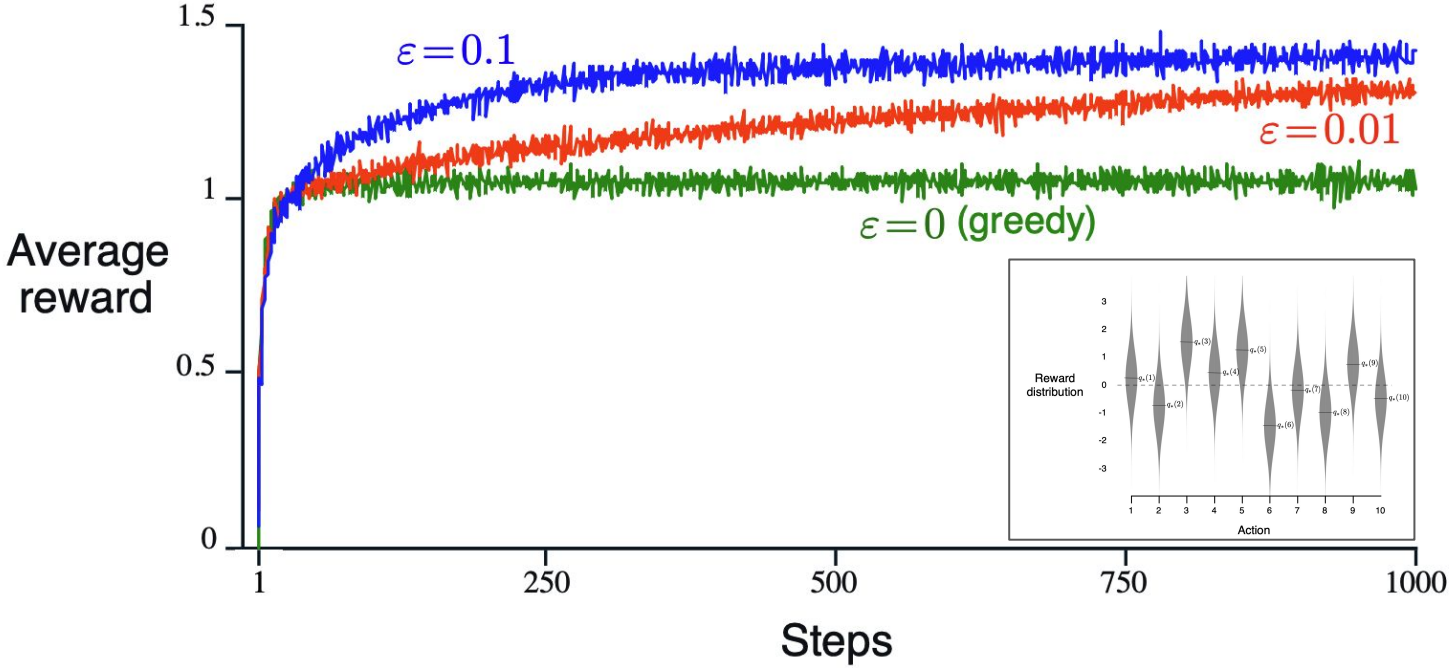
Exploration

- 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_*

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

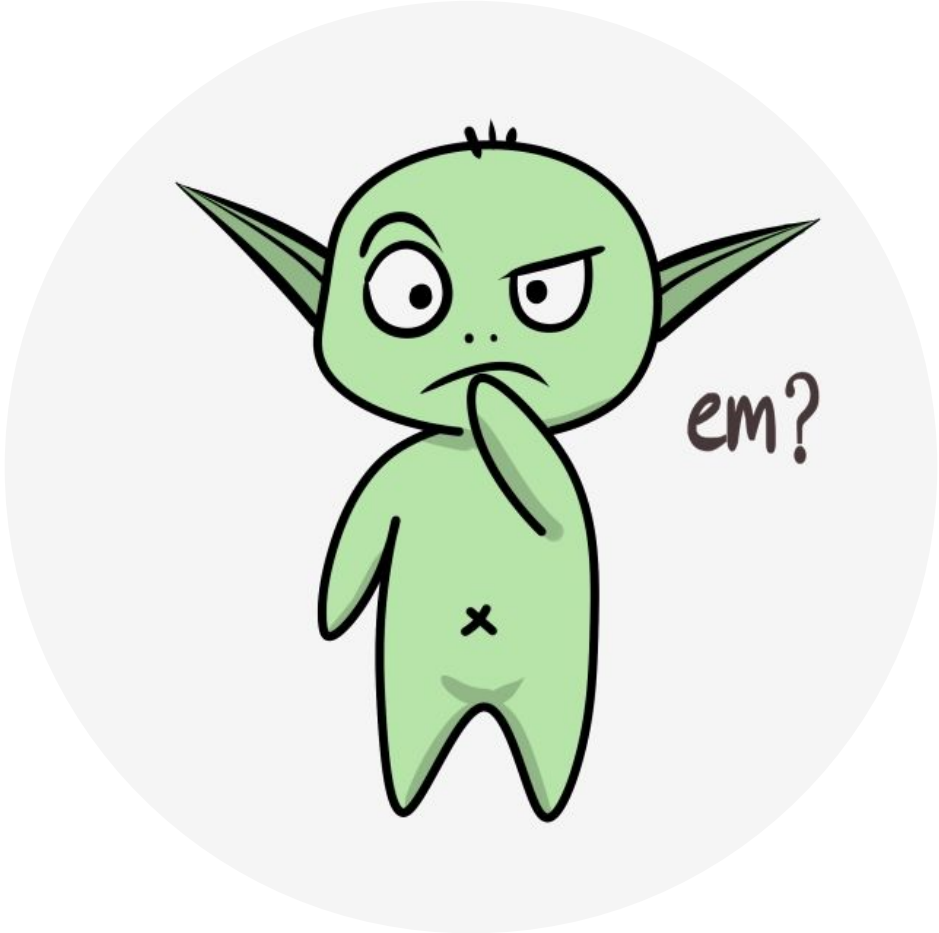
Incremental updates to estimate q_*

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

Incremental updates to estimate q_*

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

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

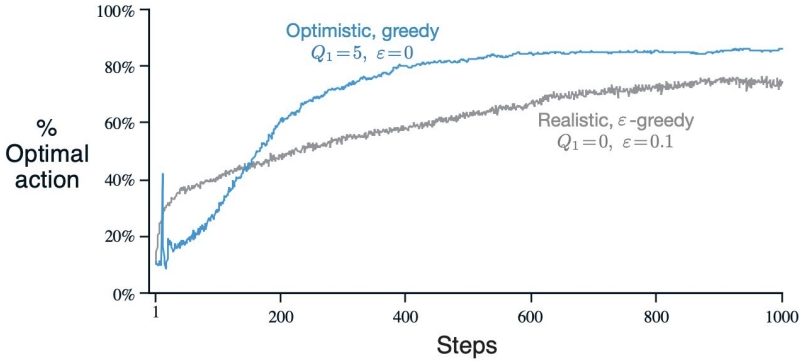
$$
\begin{aligned}
Q_{n+1} &= Q_n + \alpha\Big[R_n - Q_n\Big] \\
&= \alpha R_n + (1-\alpha)Q_n \\
&= \alpha R_n + (1-\alpha)\left[\alpha R_{n-1} + (1-\alpha)Q_{n-1}\right] \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha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

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

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).

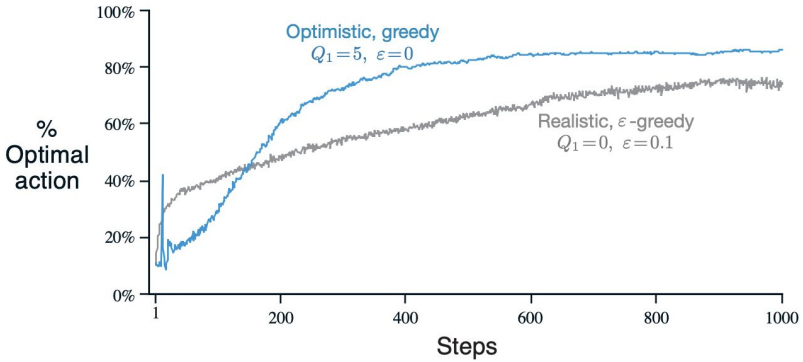


Optimism in the face of uncertainty

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

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*

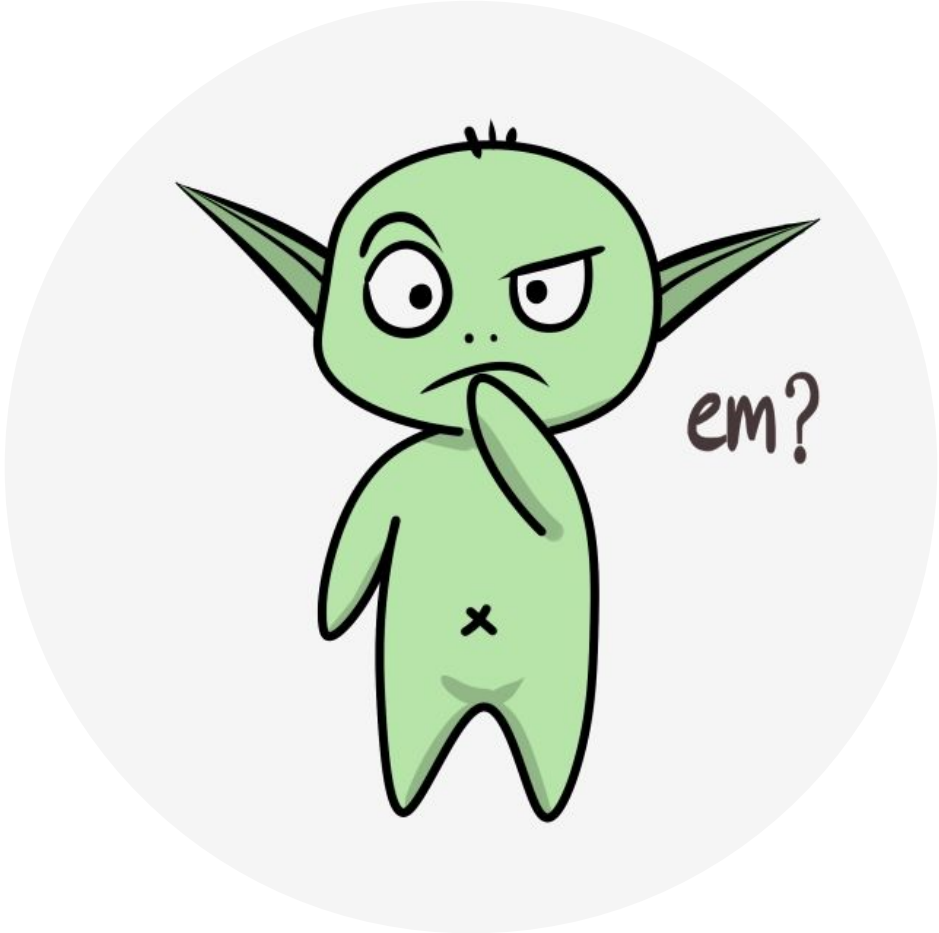
$$\left[8 \sum_{i:\mu_i < \mu^*} \left(\frac{\ln n}{\Delta_i} \right) \right] + \left(1 + \frac{\pi^2}{3} \right) \left(\sum_{j=1}^{K} \Delta_j \right)$$

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

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

Contextual bandits (Associative search)

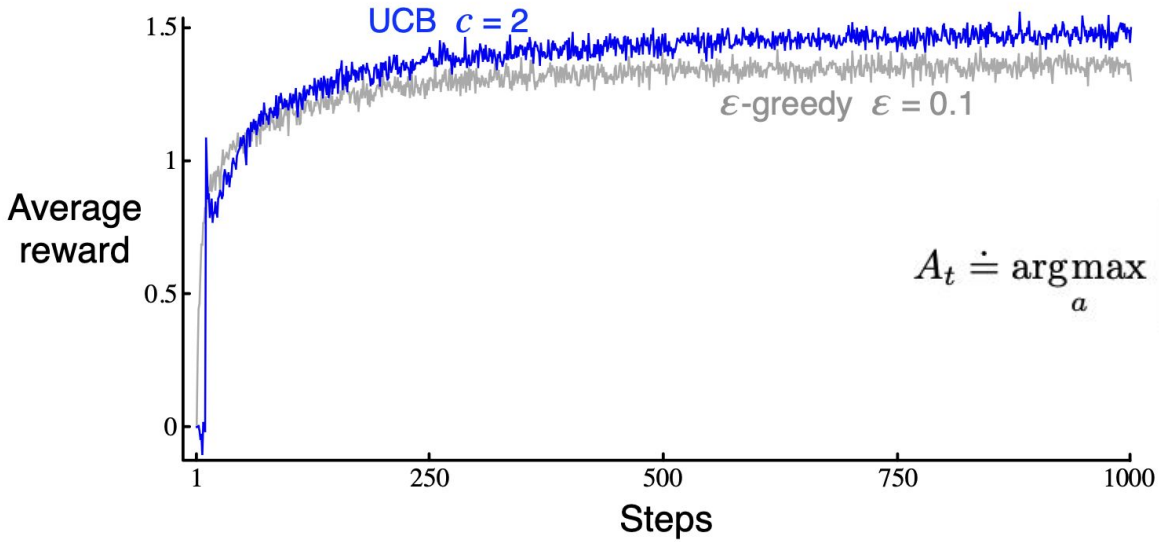
- One need to associate difference actions with different *situations*.
- You need to learn a *policy,* which is a function that maps situations to actions.
- Most real-world problems modeled as bandits problems are modeled as contextual bandits problems.
- Example: A recommendation system, which is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



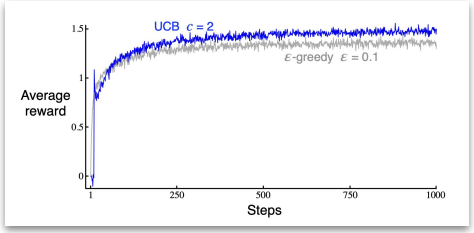
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

An exercise – Exercise 2.8 of the textbook

Exercise 2.8: UCB Spikes In Figure 2.4 the UCB algorithm shows a distinct spike in performance on the 11th step. Why is this? Note that for your answer to be fully satisfactory it must explain both why the reward increases on the 11th step and why it decreases on the subsequent steps. Hint: If $c = 1$, then the spike is less prominent. □



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



Solution – Exercise 2.8 of the textbook

We need to explain the jump in step 11 and why the smaller jump in step 12.

The jump up at step 11 is due to that being the first step on which the action selected depends on the received rewards—the first ten actions being random without replacement. It is not optimal, but definitely better than random.

At step 12, the second part of the UCB equation kicks in (because now the actions vary in the number of times they have been tried). The one action that did well in the first 10 steps, and was repeated at step 11 will now be at a disadvantage because it has been selected twice while the others only once. If c is large, then this effect dominates and the action that performed best in the first 10 steps is ruled out on step 12. If is smaller (e.g., c = 1), However, the jump down at step 12 is smaller because the advantage that comes from having been selected few times is smaller.

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

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.