Intro to RL

Jason Dolatshahi

Burlington DS Meetup

5/31

Outline

1. Introduction
2. RL Systems
3. RL Applications
4. RL Concepts
5. Python Demo

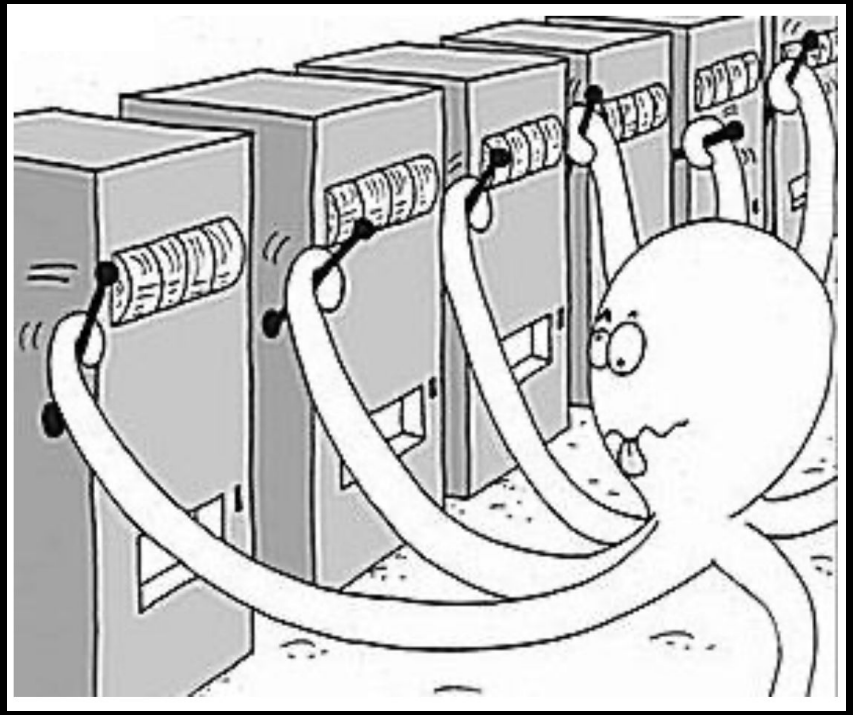
1 Introduction

Have you ever faced a decision where you had to choose between relying on what you know, or trying something new?

Imagine you're a gambler facing a row of slot machines. You want to maximize your long-term winnings, but each machine produces an unknown **reward**.

What should you do?

1 Introduction

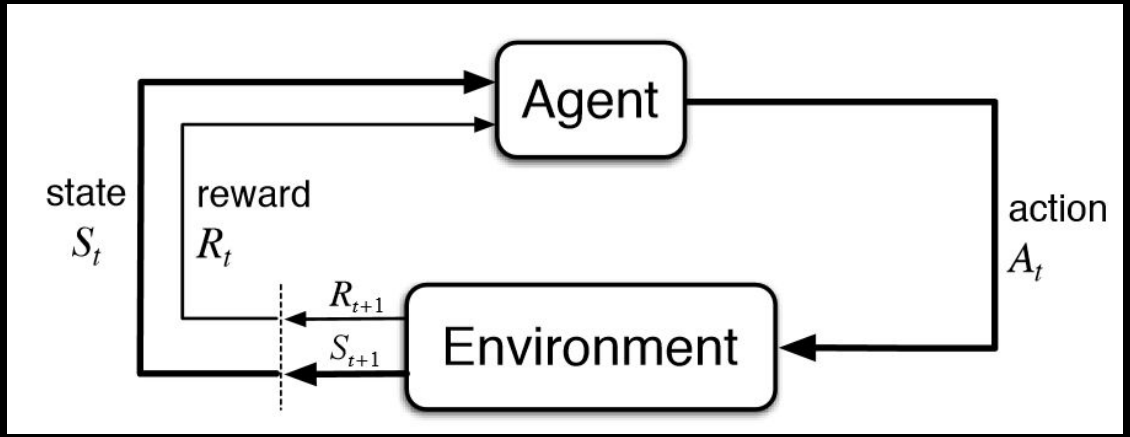


This situation is known as a **multi-armed bandit** problem.

You have *k* options (or "**arms**") to choose from, and for each trial you have to decide to either maximize expected reward, or to give up some expected reward to gain new information.

The only way to maximize long-term gains is with balance!

1 Introduction



Reinforcement learning is a powerful machine learning framework that can optimize sequential decision-making problems like this.

RL agents **learn by interacting** with their environment, unlike traditional supervised ML algorithms that require training on a historical dataset.

2 RL Systems

An **RL agent** interacts with its environment through a sequence of **repeated trials**.

The agent maximizes long-term reward by managing a **tradeoff** between exploitation and exploration.

Exploit means to maximize expected reward given current information.

Explore means to sacrifice expected reward to gain new information.

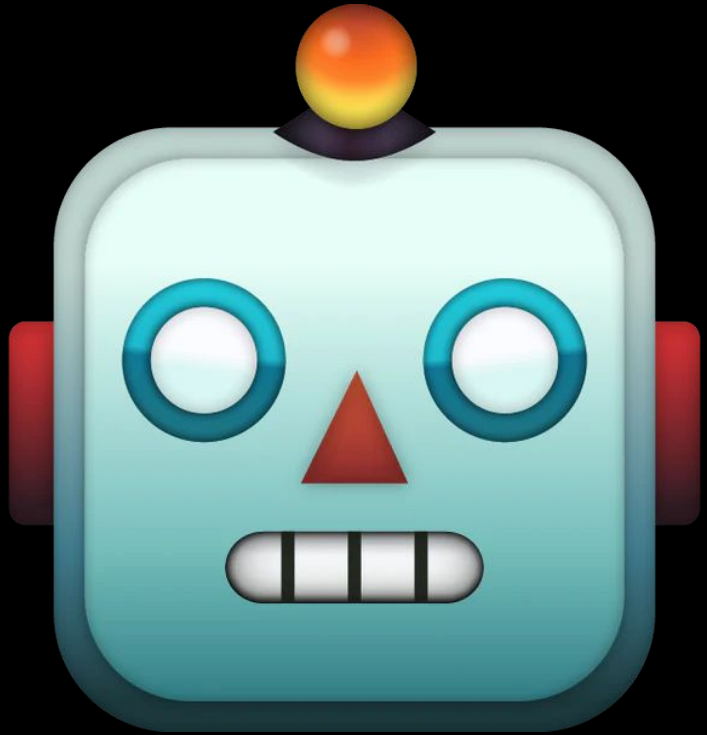
2 RL Systems

The agent **exploits** when it uses data to select an action.

This can be done, for example, by predicting expected reward with an ML model.

A lightweight agent can make predictions using just a reward signal, and more complex agents can include other information as well.

When the reward estimate depends on a set of features, the agent is called a **contextual bandit**.

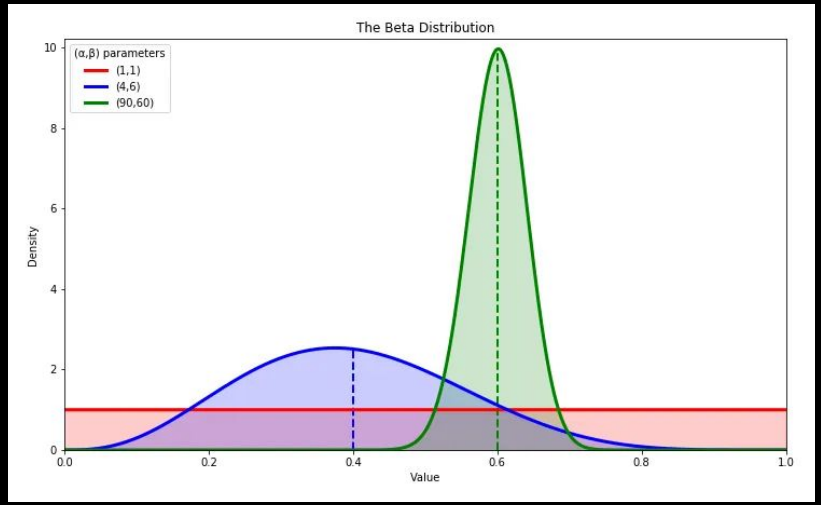


2 RL Systems

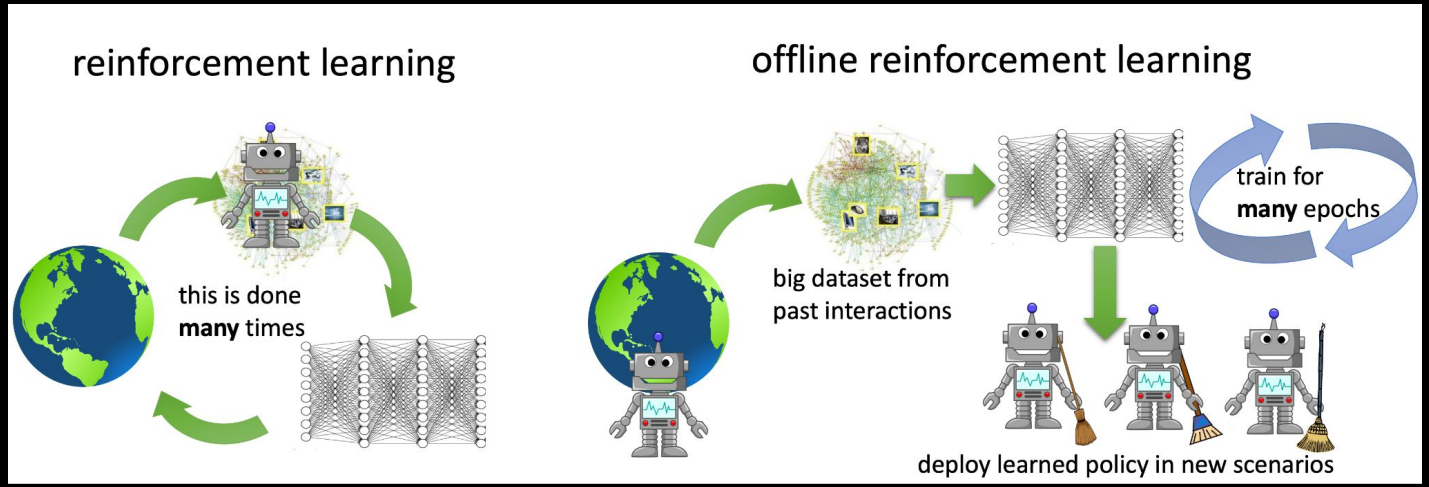
The agent **explores** to gain new information, to avoid local optima, and to adapt to changes in its environment.

One way to do this is to dedicate a fixed percentage of trials to decisions made uniformly at random. This is called **epsilon-greedy** exploration.

Another method involves balancing explore & exploit somewhat differently, by using random draws from a **rewards distribution**. We will look at a version of this called **Thompson sampling**.



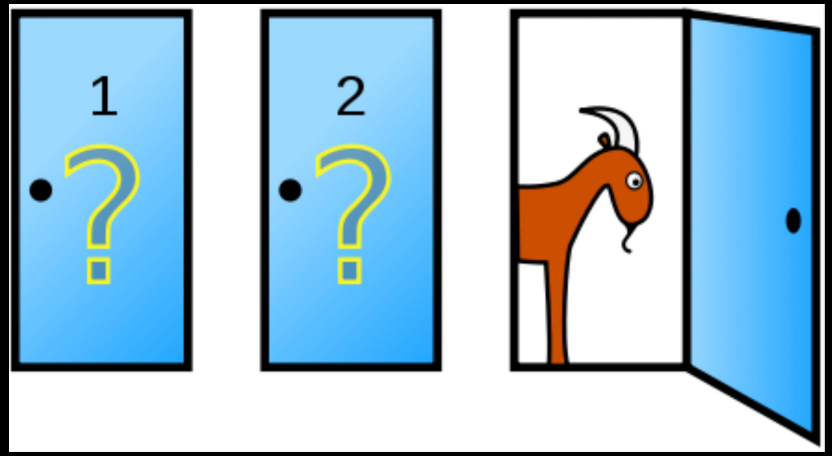
2 RL Systems



RL policy improvements can be expensive and complex to test if testing is limited to the online environment.

Using **offline optimization** to perform initial tests typically provides greater flexibility and efficiency. Offline RL involves training the agent first using a historical dataset, then deploying in an online environment.

2 RL Systems



One unique characteristic of RL is that the agent observes the reward only for its chosen action. Rewards for other actions **remain unknown**.

This is a departure from supervised ML, where all training set labels are fully revealed.

This feature of RL is called **partial information**, and it has important practical consequences!

3 RL Applications

3 RL Applications

Methods

We trained this model using Reinforcement Learning from Human Feedback (RLHF), using the same methods as InstructGPT, but with slight differences in the data collection setup. We trained an initial model using supervised fine-tuning: human AI trainers provided conversations in which they played both sides—the user and an AI assistant. We gave the trainers access to model-written suggestions to help them compose their responses. We mixed this new dialogue dataset with the InstructGPT dataset, which we transformed into a dialogue format.

To create a reward model for reinforcement learning, we needed to collect comparison data, which consisted of two or more model responses ranked by quality. To collect this data, we took conversations that AI trainers had with the chatbot. We randomly selected a model-written message, sampled several alternative completions, and had AI trainers rank them. Using these reward models, we can fine-tune the model using Proximal Policy Optimization. We performed several iterations of this process.

https://openai.com/blog/chatgpt

4 RL Concepts

policy = decision making rule (π)

context = information available (X)

action = policy choice (A)

transition matrix = probabilities for transitions between states (P)

Markov decision process (MDP) = set of all possible *(X, A, P)* plus the Markov property

Markov property = current state depends only on previous state (no path-dependence)

stationarity = homogeneity in time (important assumption in general)

regret = perfect rewards - realized rewards (goal → minimize regret)

value problem = forecast expected reward (result → value function)

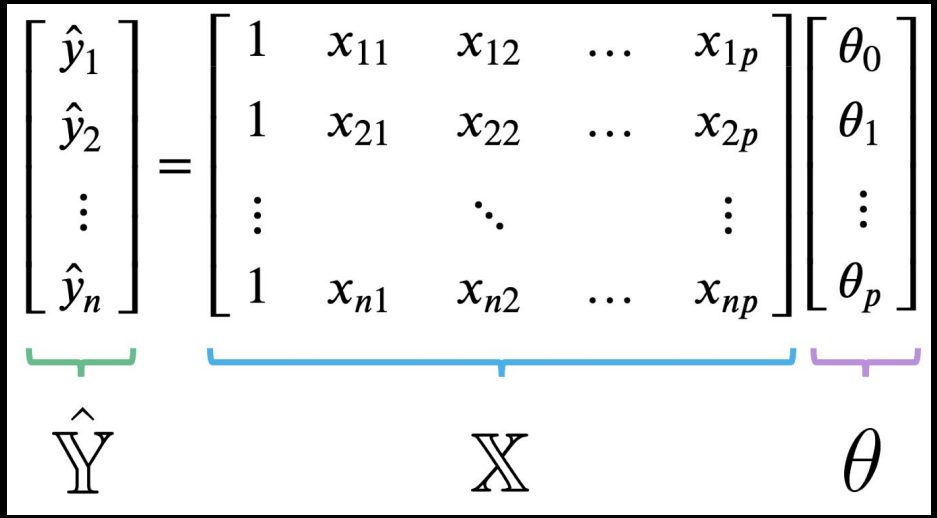
control problem = maximize expected reward (result → policy)

4 RL Concepts

RL is a powerful and widely-applicable ML technique and presents numerous opportunities for optimization, including the following:

- explore strategies
- exploit strategies
- contextual bandits
- reward shaping

Note that all of these rely on **offline optimization**!



5 Python Demo

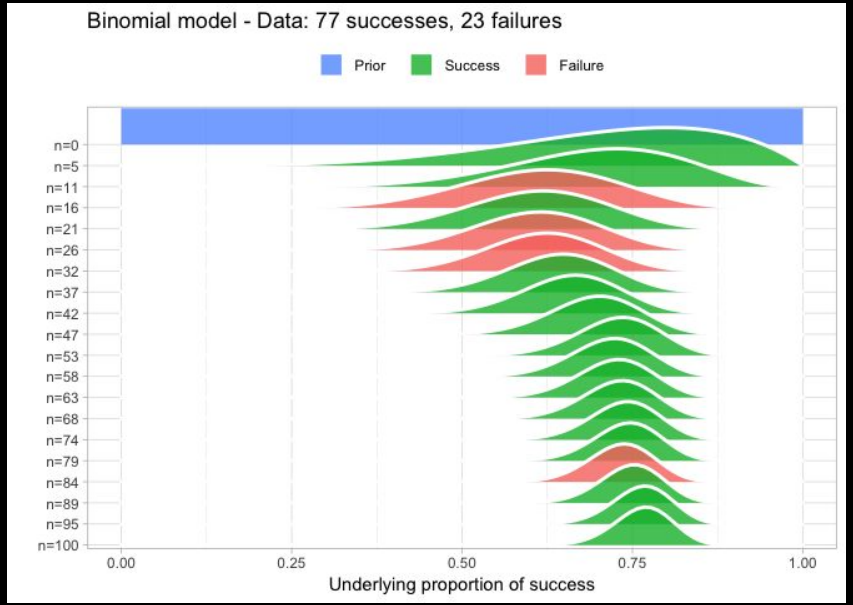
Let's take a look at some of these ideas in action.

Suppose we have a sequence of repeated trials with a binary (0/1) reward signal. Then we can use a **beta-binomial model** to estimate the success probability over a sequence of trials.

This is a **Bayesian** model that works by using a **binomial** distribution to model the reward signal, and a **beta** distribution to characterize our prior knowledge.

In an RL context, this is called **Thompson sampling**.

http://localhost:8888/notebooks/Quick%20RL.ipynb



References

Books:

http://incompleteideas.net/book/the-book-2nd.html
https://sites.ualberta.ca/~szepesva/rlbook.html

Articles & blog posts:

https://ai.googleblog.com/2016/01/alphago-mastering-ancient-game-of-go.html
https://arxiv.org/abs/1707.02038
https://arxiv.org/abs/1003.0146
https://bair.berkeley.edu/blog/2020/12/07/offline/
https://www.cs.mcgill.ca/~dprecup/courses/AI/Lectures/ai-lecture15.pdf
https://www.deepmind.com/learning-resources/introduction-to-reinforcement-learning-with-david-silver
https://huggingface.co/blog/rlhf
https://www.technologyreview.com/2022/05/27/1052826/ai-reinforcement-learning-self-driving-cars-autonomous-vehicles-wayve-waabi-cruise/
https://towardsdatascience.com/thompson-sampling-fc28817eacb8