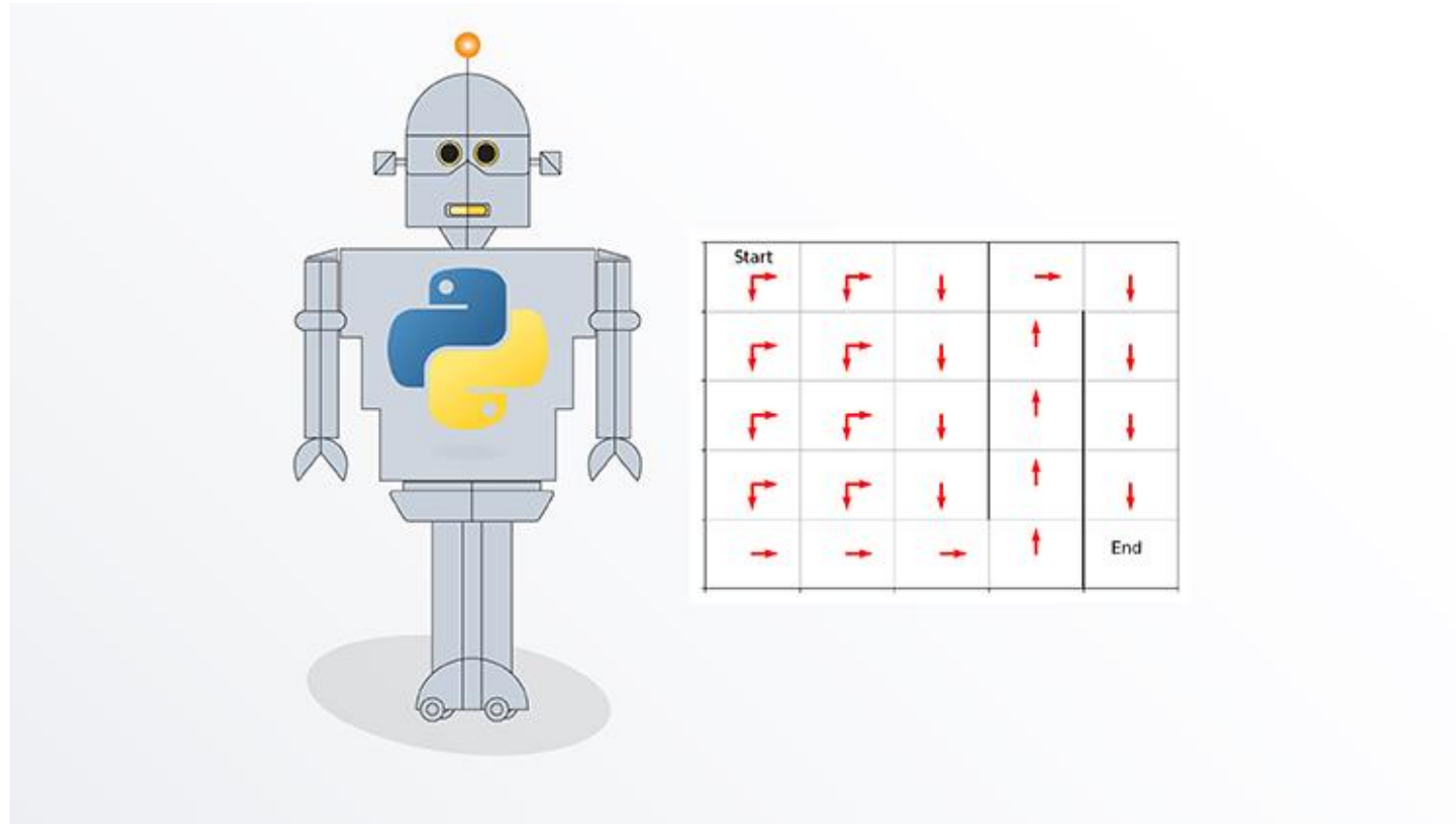


What is Machine Learning?

Chapter 4: Reinforcement Learning



Reinforcement Learning

Definition (see Wikipedia page on Reinforcement Learning)

- Area of machine learning concerned with how software agents should take actions in an environment to maximize the cumulative reward

Informal Definition:

- Learn strategy to maximize the cumulative reward

K Bandit Problem

- One-armed bandit is a nickname for a slot machine
- K Bandit problem:
 - K slot machines $n=1,2,\dots,K$ each with own mean M_n and payoffs drawn from a normal distribution
 - If means are not known, what strategy to follow to maximize cumulative payoff after many pulls?



By Yamaguchi先生, CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=57295504>

K Bandit Problem

Optimal strategy to get greatest cumulative reward: pull lever for machine with largest mean reward

Agent does not know which machine has largest mean reward -> needs to learn

Greedy Strategy:

- Track sample mean reward for each slot machine
- Pull lever for slot machine with largest current sample mean

Consider example with 2 slot machines

- Machine 1 has actual mean reward = 1
- Machine 2 has actual mean reward = 2

Action	Reward	Cumulative Reward	Machine 1 # Pulls	Machine 1 Total Reward	Machine 1 Sample Mean	Machine 2 # Pulls	Machine 2 Total Reward	Machine 2 Sample Mean
Initial	0	0	0	0	0	0	0	0
Random: Pull M1	1	1	1	1	1	0	0	0
Greedy: Pull M1	-2	-1	2	-1	-0.5	0	0	0
Greedy: Pull M2	1	0	2	-1	-0.5	1	1	1
Greedy: Pull M2	3	3	2	-1	-0.5	2	4	2

Exploit versus Explore in Reinforcement Learning

Issue with Greedy Approach:

- May get stuck pulling lever of machine with lower mean
- Leads to Exploit versus Explore dilemma

Exploit:

- At current position, take action to maximize immediate reward

Explore:

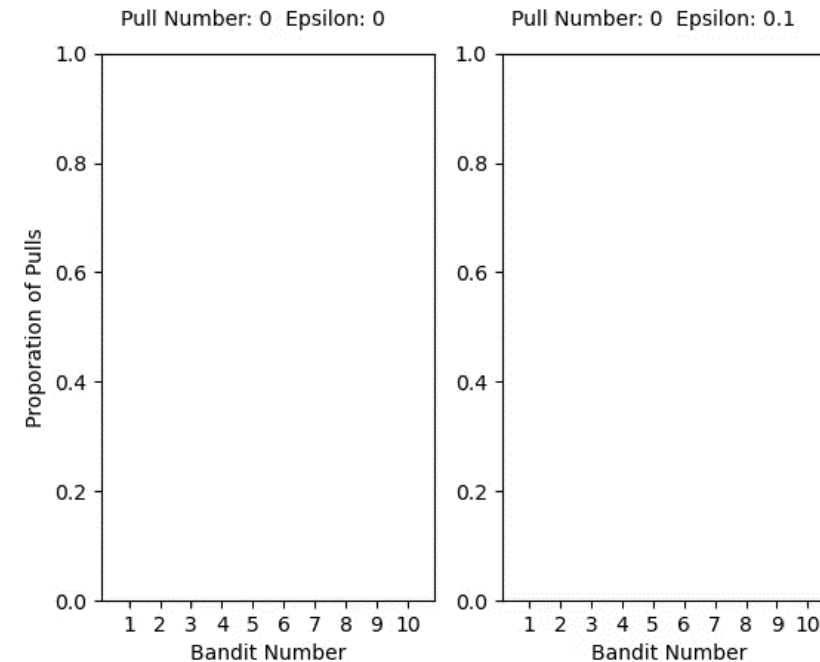
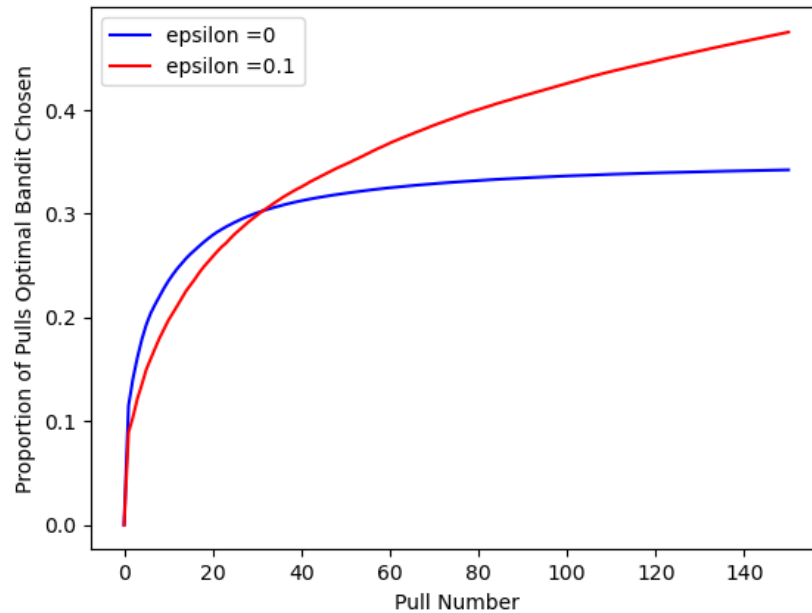
- At current position, take action to learn about environment
- Information acquired in exploration to be used later to hopefully increase cumulative reward

Epsilon Greedy Strategy:

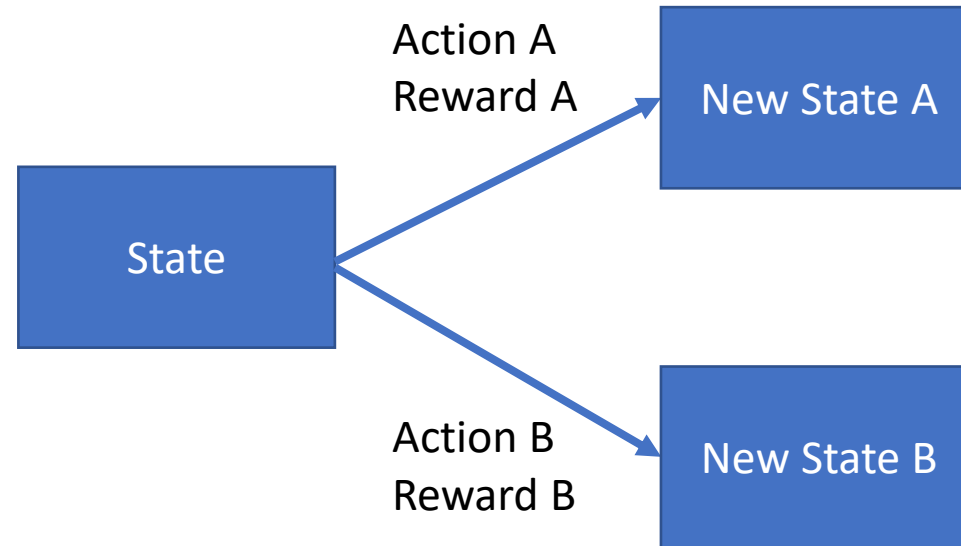
- For ϵ proportion of time pull lever of slot machine at random
- For $1 - \epsilon$ proportion of time pull lever of machine with largest current sample mean

K Bandit Problem - Simulation

- 10 slot machines
- Means drawn from standard normal distribution
- Results averaged over 1000 simulations
- Compare Greedy and Epsilon Greedy Approach with $\epsilon = 0.1$



General Formulation for Reinforcement Learning

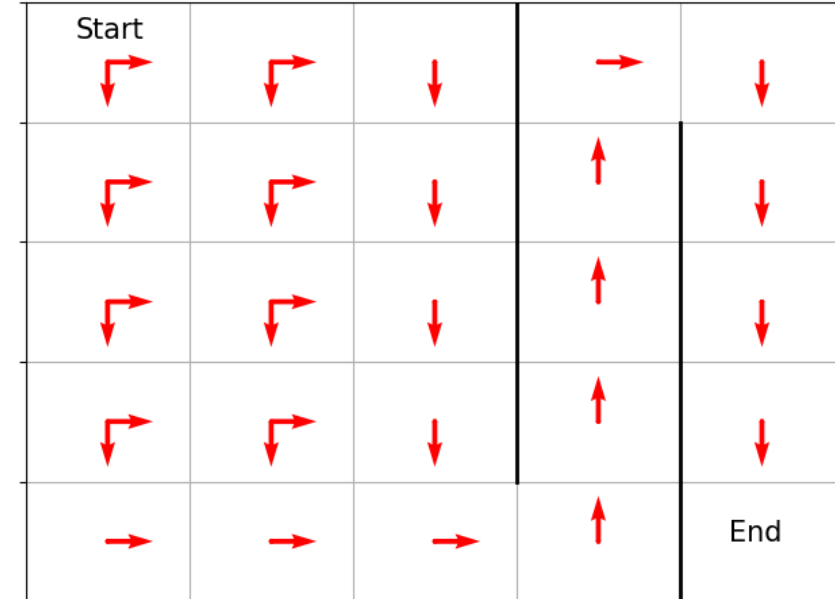
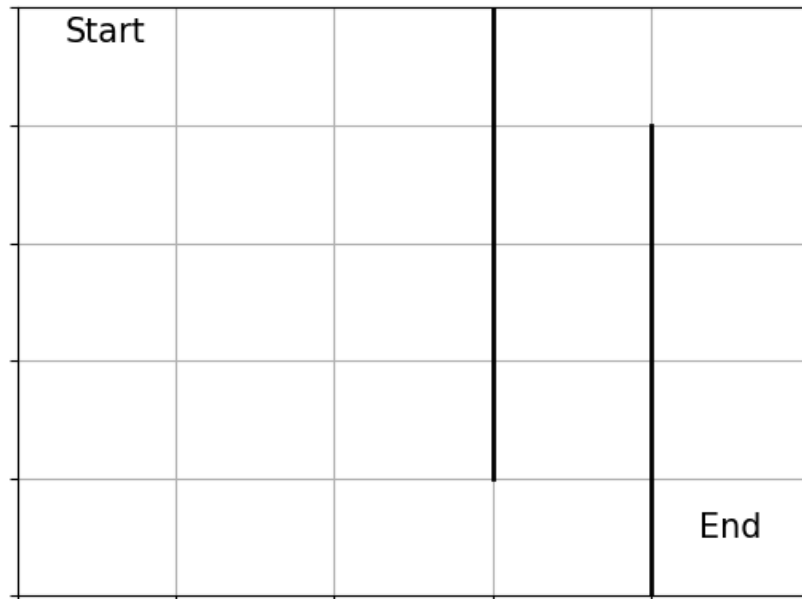


Starting at State

- Agent is allowed to take a number of possible actions, each with a specified reward and leading to a new state, where process is repeated
- Process continues forever or until reaching a terminal state
- Goal: find optimal strategy (action at each state) to maximize total cumulative reward (which may involve discounting)

Maze Problem

- State is location in maze
- Actions: up, down, left or right (stay in same place if action means hitting a wall)
- Reward is -1 for each step (including if hit wall)
- Cumulative reward is $(-1) * \text{number of step to go to End}$
- Goal: find strategy that maximizes cumulative reward from Start to End (equivalent to minimizing number of steps)

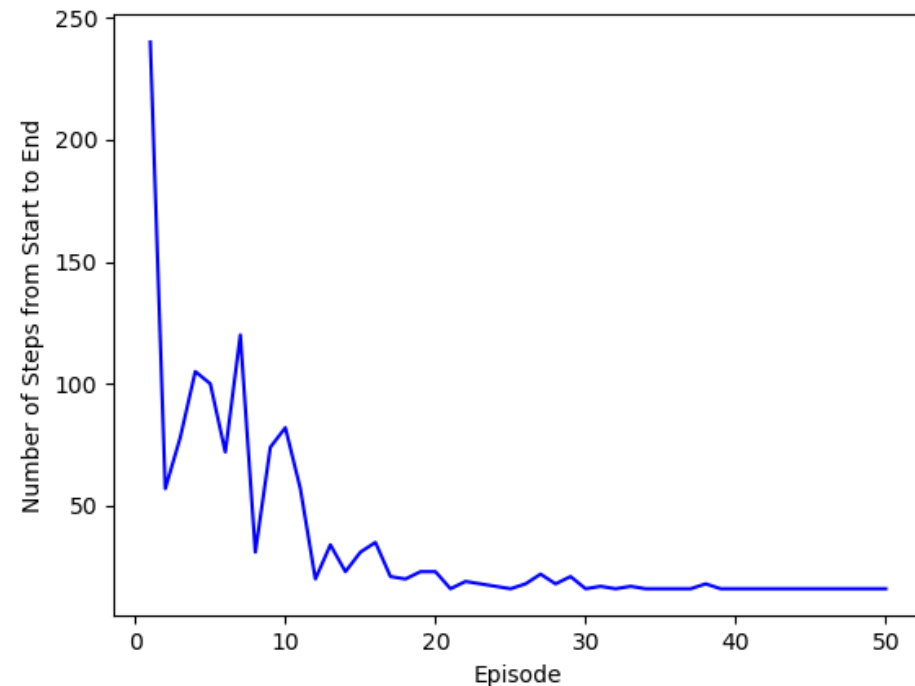
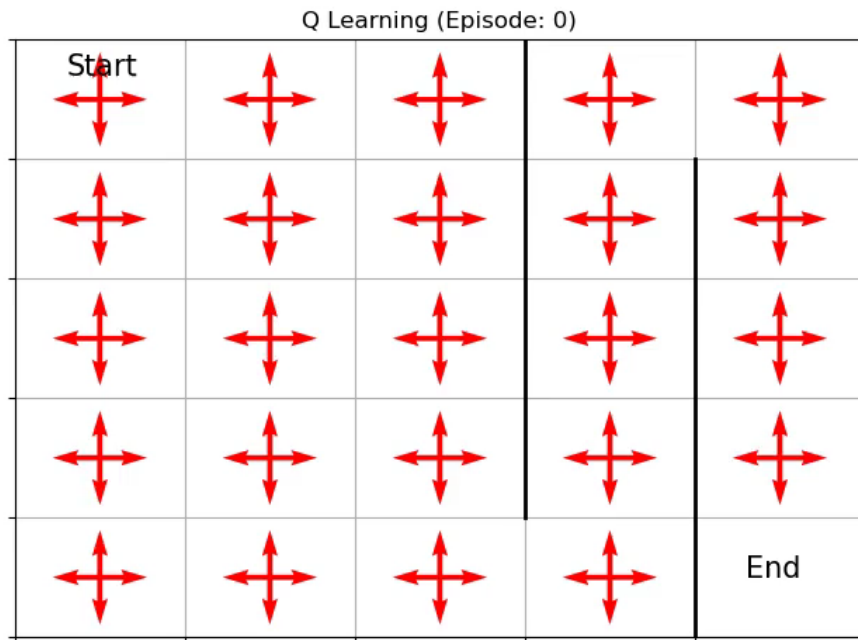


Q Learning Approach to Finding Path

- Use “Q Learning” approach
 - Simulate “episode” (sequence state/action/state/action... to go from Start to Finish)
 - Start with equal probability for up, down, left, or right actions at each state
 - Update “action-value” function $Q(s,a)$ which tracks cumulative reward for each state/action pair
 - Follow greedy strategy -> take action that maximizes cumulative reward
 - Repeat for many episodes
 - This process yields optimal strategy, obtained as maximum of action-value function over all possible actions at each state

Maze: Results from Q Learning

- Animation shows how strategy evolves from initial equally likely actions to optimal actions (actions updated after each episode)
- Plot shows Number of Moves from Start to End as a function of episode
 - Settles down at 16 steps = minimum number of steps from Start to End



Reinforcement Learning: Applications

Application	Notes
Industrial Control	States: location on factory floor, battery status Actions: perform tasks or return to home base for charging Rewards: positive rewards for performing task, negative reward for running out of power in middle of factory floor
Game Playing	States: configuration of pieces on the game board Actions: allowable moves Rewards: typically positive reward for winning game (may be additional intermediate rewards for other accomplishments during play)

Reinforcement Learning: Notes

Component	Notes:
Q Learning and related algorithms	<ul style="list-style-type: none">• Suitable if reasonable number of states and actions:• Maze problem has 24 states/4 actions per state -> 96 action values to learn
Dyna Q	<ul style="list-style-type: none">• Can reduce number of episodes to learn optimal strategy by incorporating planning as in Dyna Q algorithm
	<p>Must use alternative approaches if there are many states/actions</p> <ul style="list-style-type: none">• Chess has astronomical number of configurations of pieces on the board• Use approximation methods to estimate action values for state/action pairs (can use ideas from Supervised Learning)
AlphaZero	<p>Effort to create artificial intelligence program to play chess, shogi and go (see AlphaZero Wikipedia page and see AlphaGo movie on Youtube)</p> <ul style="list-style-type: none">• Trained using self play only• Beat other state of the art programs in chess, go, shogi