# Introduction:

Dec 21, 2021

In reinforcement learning, the reward gives the agent some idea of how good/bad its recent actions were.

Supervised Learning → Teacher tells you correct answer.

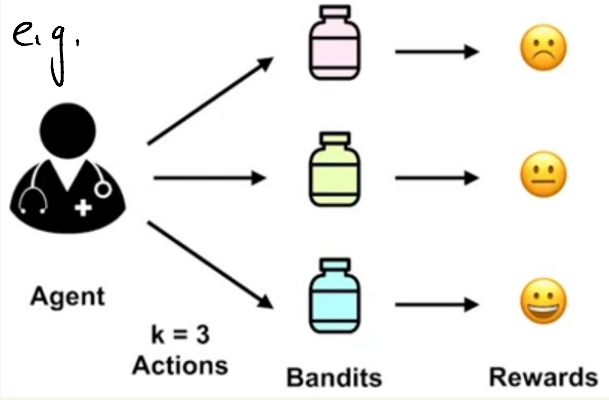
RL → Teacher identifies good behavior but can't tell you exactly how to do it.

Unsupervised → Used to find underlying structures in data (not really related)

- Both supervised and unsupervised learning can be used w/ RL to help w/ generalization.
- RL uses online learning, which is different than learning from data.

# The k-armed Bandit Problem:

We have an agent who chooses between k actions and receives a reward based on the action it chooses.




# Action-Value Function:

Helps agent decide what action to take.

$$q_*(a) \equiv \mathbb{E}[R_t \mid A_t = a] \; \forall a \in \{1 \dots k\}$$

$t$: time step

$A_t$: action selected at time step $t$

$R_t$: corresponding reward (not equal to expected reward)

$a$: arbitrary action

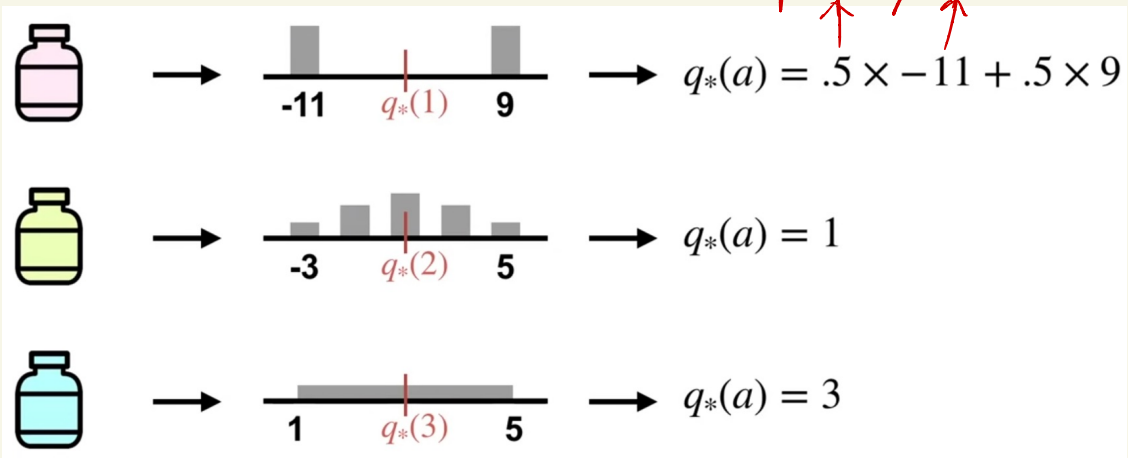
$q_*(a)$: expected reward (aka. action value) from choosing $a$

The value is the expected reward ($q_*(a)$).
The goal is to maximize the expected reward.

↳ $\underset{a}{\text{argmax}}\, q_*(a)$ # Action $a$ which maximizes $q_*(a)$

# Rewards are given based on fixed probability distributions.

## Calculating $q_*(a)$:



# Chapter 2 - Multi-Armed Bandits:

RL uses purely evaluative feedback rather than instructive feedback. It evaluates how good an action was rather than whether it was the best/worst possible action.

→ Supervised learning indicates correct action, independent of actual action taken

→ RL is dependent on action taken

## 2.1 - A k-armed Bandit Problem:

Scenario: Make a choice between k different actions. You receive a reward after each action based on a stationary probability distribution.

※ i.e. The reward won't be the same each time, but one action that rewards more will always reward more.

Goal: Maximize expected total reward over 1000 time steps (action selections). ※ You choose

Strategy: Learn which action is the best and keep choosing it.

$q_*(a) \equiv E[R_t | A_t = a]$ → expectation of $R_t$ given we selected action $a$

$t$: time step

$A_t$: action selected at time step $t$

$R_t$: corresponding reward

$a$: arbitrary action

$q_*(a)$: expected reward from choosing $a$

Greedy Action: Selecting an action that has the greatest estimated value.

↳ Exploitation maximizes reward on one step.

Nongreedy Action: Exploring other actions to improve your estimate of their rewards.

↳ Exploration may produce greater total reward in the long run (after finding an action w/ greater reward, etc).

⚠ To balance the two is complicated, based on values of estimates, uncertainties, and # of remaining steps.

# 2.2 - Action-Value Methods:

Methods for estimating values of actions and using these estimates to make action selection decisions.

$$Q_t(a) \equiv \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\#\text{ times } a \text{ taken prior to } t}$$

$Q_t(a)$: estimated value of action $a$ at time step $t$

↳ $$Q_t(a) \equiv \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{I}_{A_i = a}}{\sum_{i=1}^{t-1} \mathbb{I}_{A_i = a}}$$

✡ Sample-Average Method

✡ $\mathbb{I} = 1$ if $A_i = a$, else $0$, s.t. $Q_t(a)$ only depends on when $a$ was chosen.

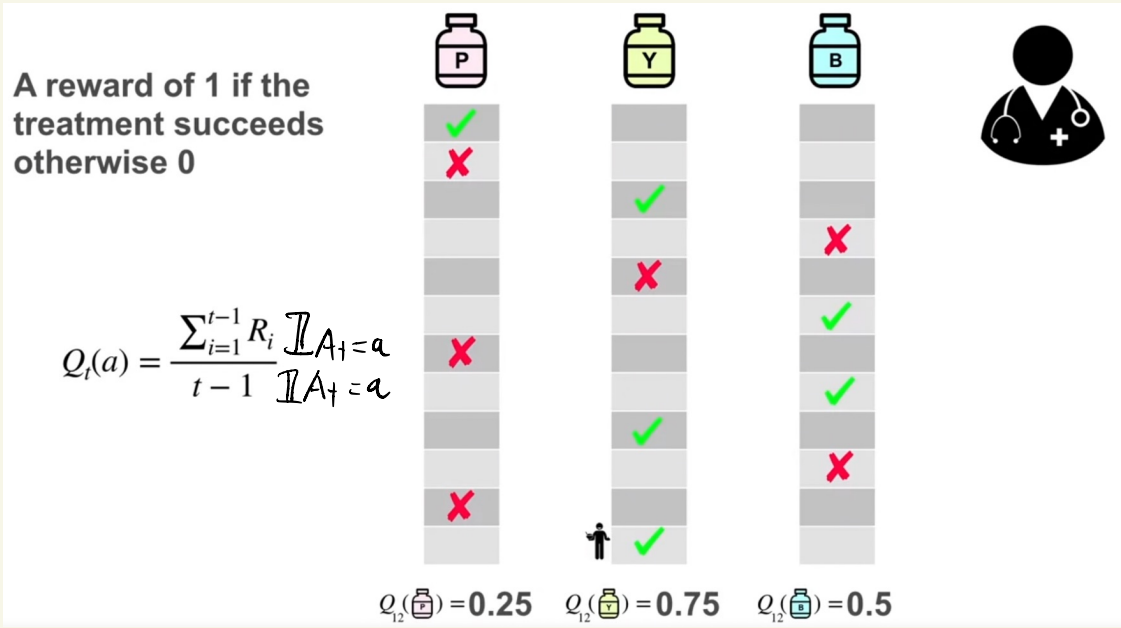
✡ If denominator is $0$ initially, $Q_t(a) = 0$.

$$A_t \equiv \underset{a}{\operatorname{argmax}}\, Q_t(a)$$

✡ Greedy Action Selection Method

✡ Choose $A_t = a$ that maximizes $Q_t(a)$.

e.g. Sample-average method




## 2.4 - Incremental Implementation:

$$Q_n \equiv \frac{R_1 + R_2 + \dots R_{n-1}}{n-1}$$

How do we implement this?

$R_i$: reward received after $i^{th}$ selection of this action

$Q_n$: estimate of value after action has been selected $n-1$ times

✌ Use incremental formulas to reduce memory and computation requirements.

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

✌ Requires only memory for $Q_n$ and $n$, instead of $R_1 \dots R_n$ and $n$.

$\therefore$ $\boxed{Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]}$ ✌ Incremental Update Rule

NewEstimate ← OldEstimate + StepSize [Target − OldEstimate]

✌ Doesn't this look familiar? (Weights update)

e.g. $\varepsilon$-greedy action algorithm

**A simple bandit algorithm**

Initialize, for $a = 1$ to $k$:
$Q(a) \leftarrow 0$
$N(a) \leftarrow 0$

Loop forever:
$A \leftarrow \begin{cases} \text{argmax}_a\, Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$
$R \leftarrow bandit(A)$
$N(A) \leftarrow N(A) + 1$
$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}[R - Q(A)]$

# 2.5 - Tracking a Nonstationary Problem

Reward probabilities may change over time.

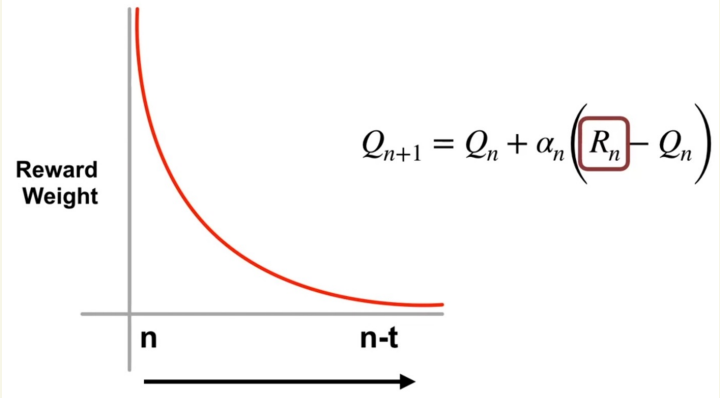
↳ Give more weight to recent rewards

↳ Use a constant stepsize parameter $\alpha$

$$Q_{n+1} \doteq Q_n + \alpha[R_n + Q_n], \quad \alpha \in (0, 1] \text{ is constant}$$

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

The weight decays exponentially




$$\therefore Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i$$

$Q_1$: initial action-value

↳ $\lim_{n\to\infty} (1-\alpha)^n Q_1 = 0$

## Exploration and Exploitation:

Exploration: Improves knowledge for long-term benefit.

↳ Choose wide variety of actions to discover the best one.

Exploitation: Exploit knowledge for short-term benefit.

↳ Based on one/a few trials, keep choosing the best one found.

## Exploration/Exploitation Tradeoff:

The agent wants to explore to get more accurate estimates of its values. It also wants to exploit to get more reward, but it cannot do both simultaneously.

## Epsilon-Greedy Action Selection:

$\varepsilon$: chance to explore

$$A_t \leftarrow \begin{cases} \underset{a}{\operatorname{argmax}}\, Q_t(a) & \text{w.p. } 1-\varepsilon \\ a \sim \text{Uniform}(\{a_1, \ldots a_k\}) & \text{w.p. } \varepsilon \end{cases}$$

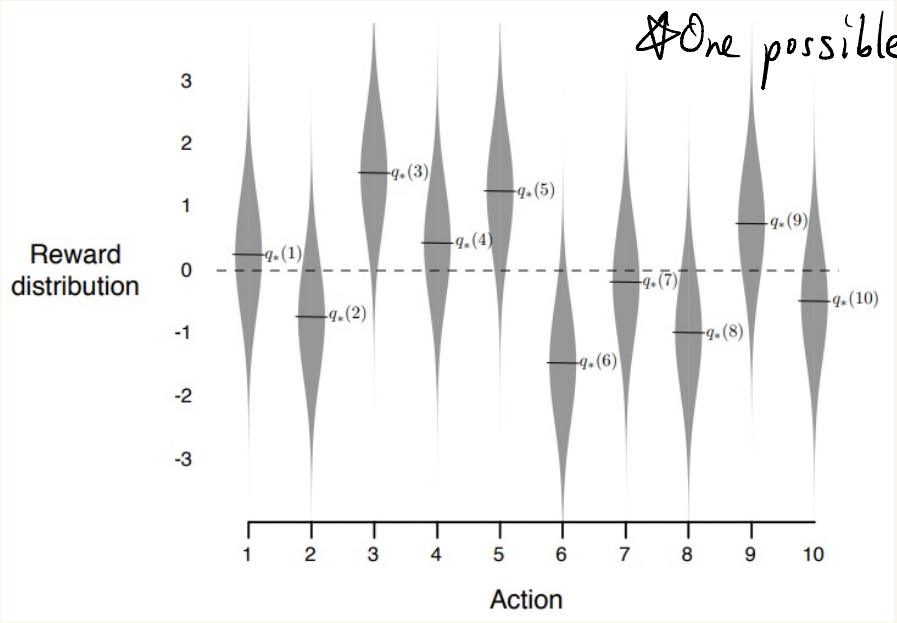
w.p. = with probability

Epsilon-Greedy Action Selection

Let's see this effect on the 10-armed Testbed

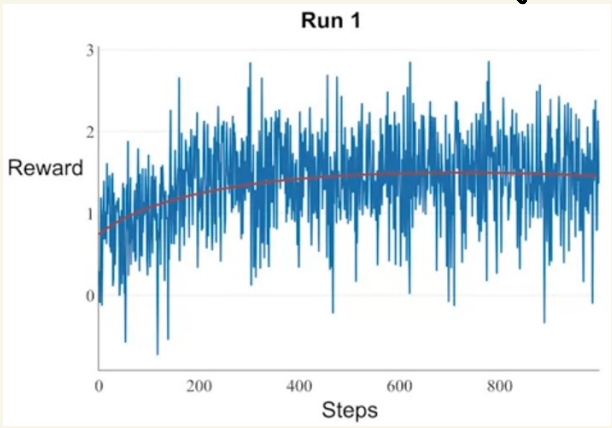
# 2.3-The 10-armed Testbed!

A set of 2000 randomly-generated k-armed bandit problems w/ k=10.

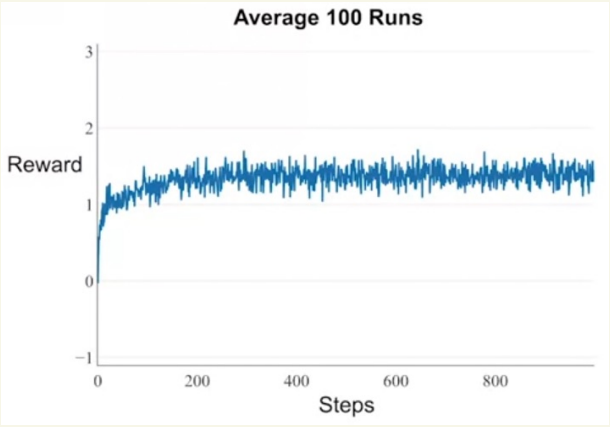



☆One possible problem

Each learning method was measured over 1000 time steps for each of the 2000 runs. Its performance was the avg over the 2000 runs.

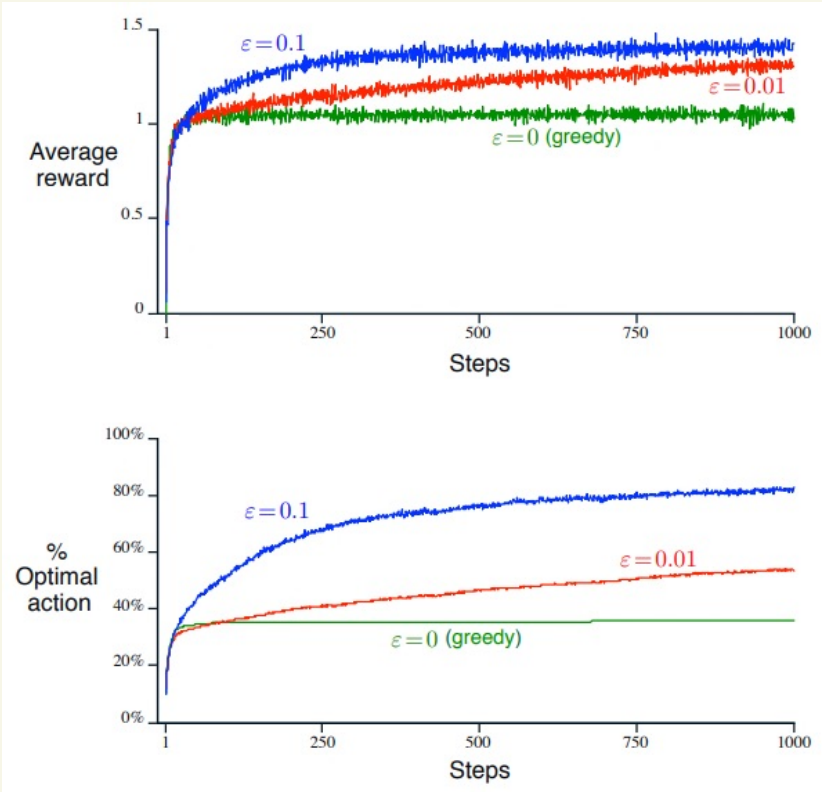



☆For epsilon-greedy




*Less spikes w/ more runs.

## Final Results:



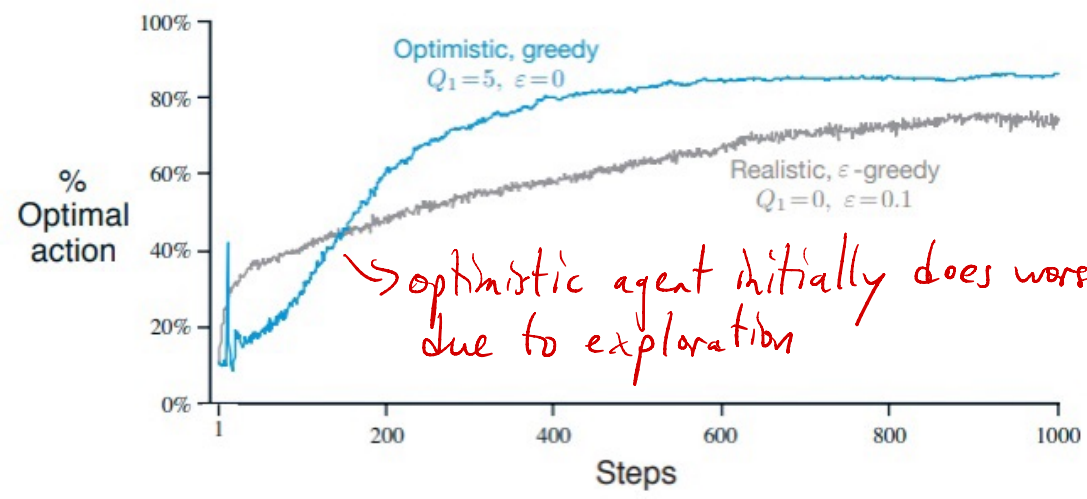

*Notice that greedy does good w/ less steps

# 2.6 - Optimistic Initial Values:

A technique used to encourage exploration.

↳ Set $Q_1(a) = +5$, w/ $q_*(a)$ having mean = 0, variance = 1.

Agent will initially be disappointed by any action it chooses and keep exploring. All actions are then tried several times until value estimates converge.

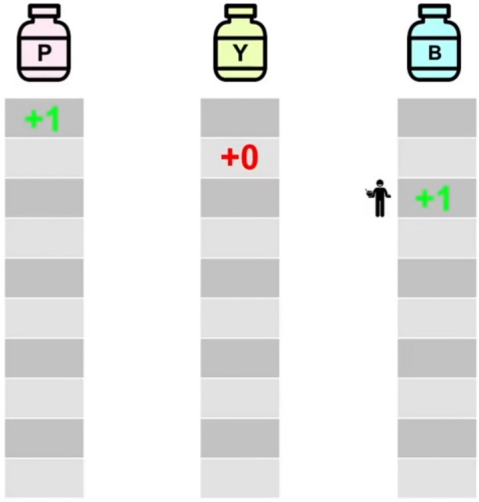



↳ Optimistic agent initially does worse due to exploration

* $Q_1 = 5$'s effect decreases over time, so exploration only encouraged early on.

e.g. $Q_1 = +2$

A reward of 1 if the treatment succeeds otherwise 0




$$Q_{n+1} \leftarrow Q_n + \alpha\left(R_n - Q_n\right)$$

**Let** $\alpha = 0.5$

$Q_2 \leftarrow 2 + 0.5(1-2) = 1.5$

Even if treatment is correct, reward still goes down cuz $Q_1 = +2$, ∴ keep exploring

$Q_4(P) = 1.5$ $Q_4(Y) = 1.0$ $Q_4(B) = 1.5$

$q_*(P) = 0.25$ $q_*(Y) = 0.75$ $q_*(B) = 0.5$

Limitations:

1. Only encourages exploration early on.
2. Not great for non-stationary problems (changing reward distribution).
3. Hard to determine what optimistic initial value should be.

# 2.7 - Upper-Confidence-Bound Action Selection:

We want to account for:

1. How close an action is estimated to produce the largest reward.
2. The uncertainty in the estimate.

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

# UCB Action Selection

exploitation

exploration

$N_t(a)$: # times $a$ has been chosen prior to $t$

$c > 0$: controls degree of exploration

$\sqrt{\frac{\ln t}{N_t(a)}}$: uncertainty term