Multi-armed Bandits

Thomas Moerland

Course: Reinforcement Learning
Bachelor Artificial Intelligence, Leiden University

Content

1. Bandit definition

2. Exploration/exploitation

3. Updating a mean

Break

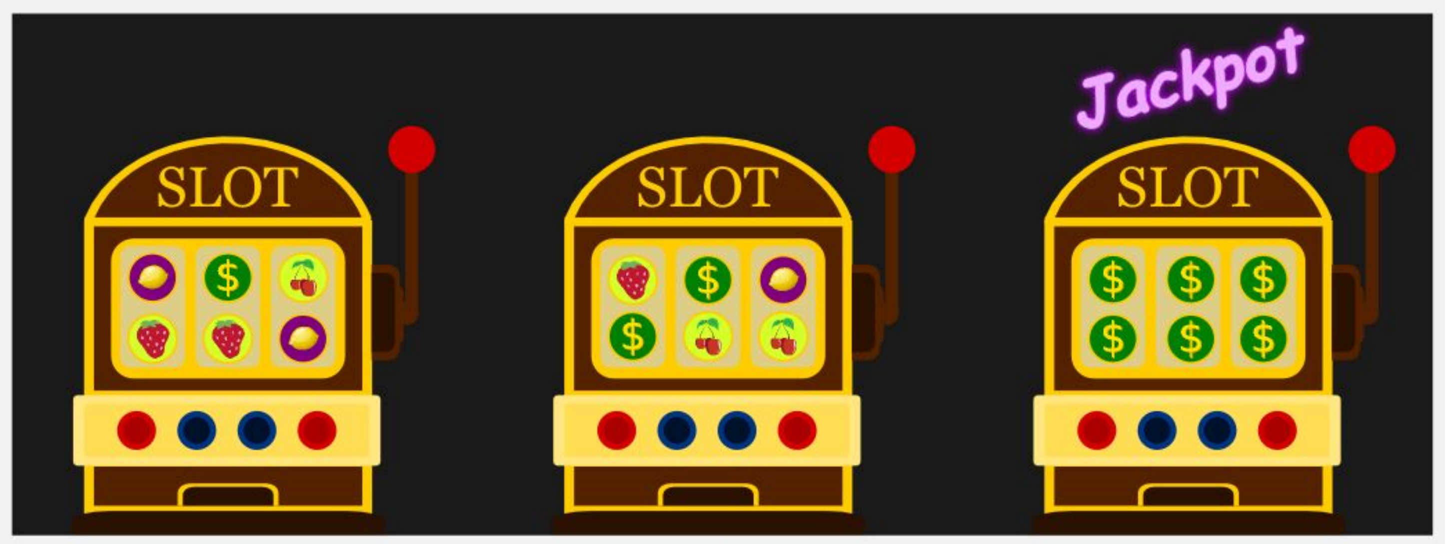
4. Bandit algorithms
    a. Random perturbation
    b. Optimistic initialization
    c. Optimism in the face of uncertainty

5. Contextual bandits & MDPs

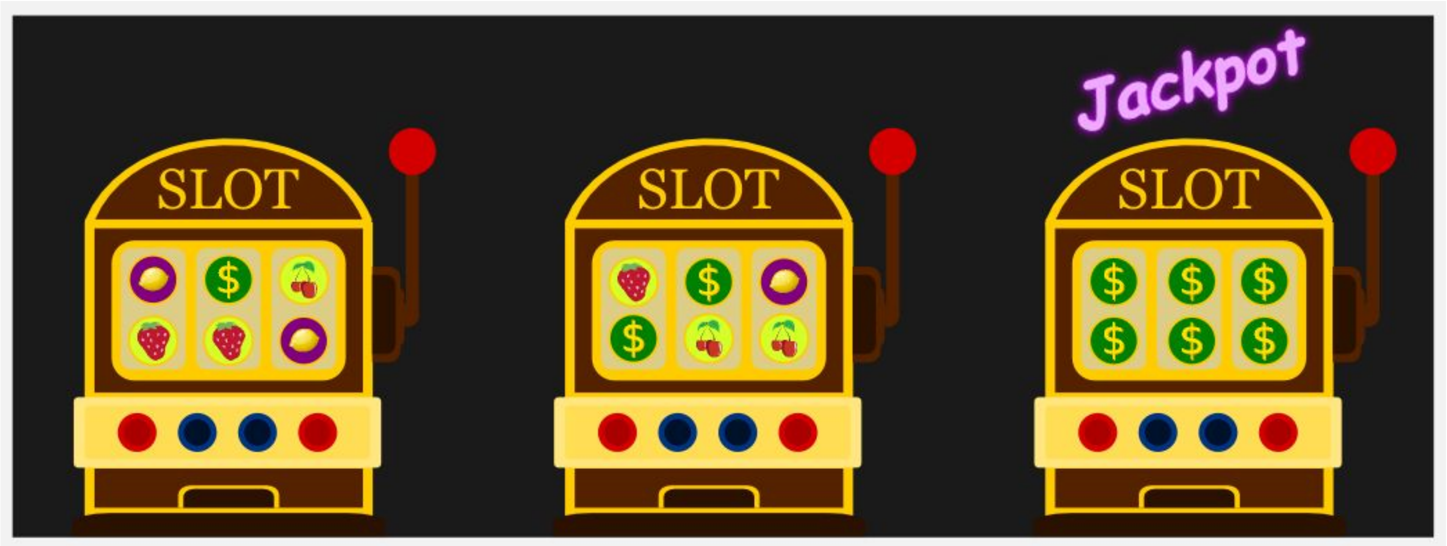
Part 1:

Bandit definition

Multi-armed bandit

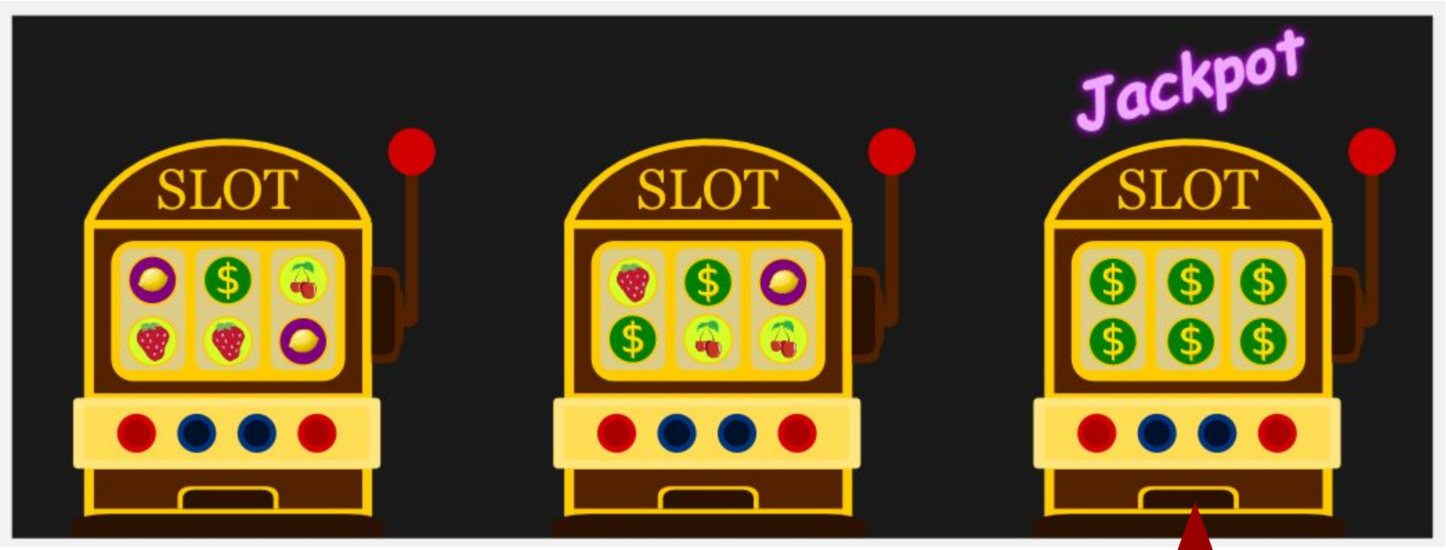


Multi-armed bandit



Don't know the pay-off of each arm/action

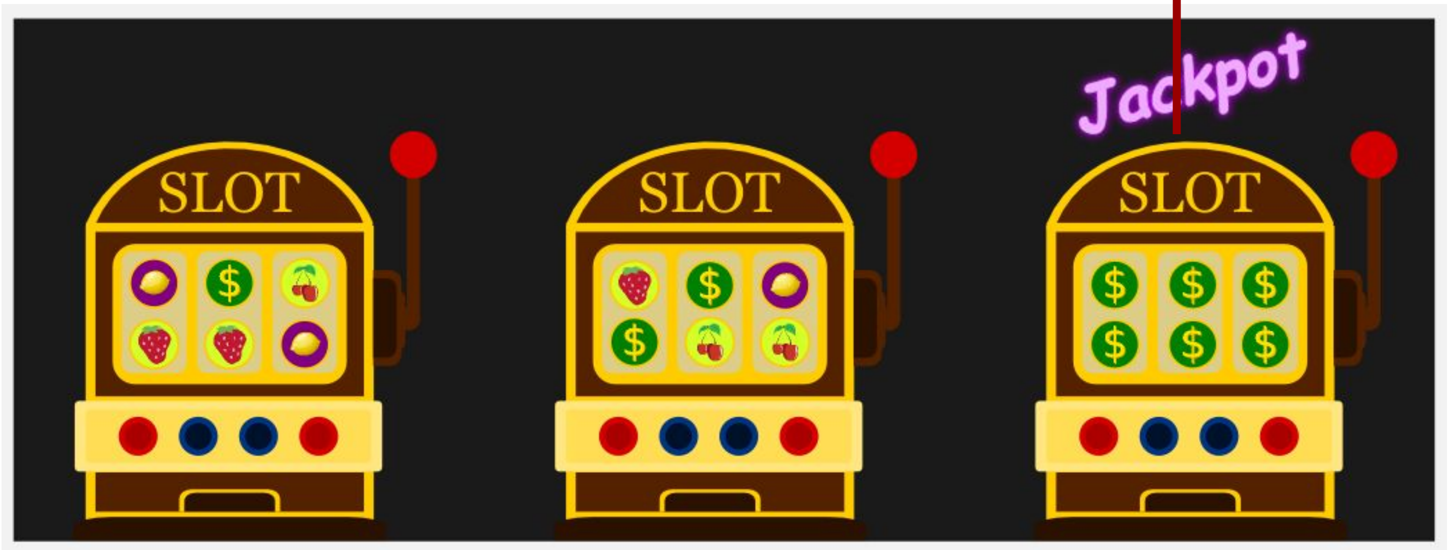
Multi-armed bandit



Let's try an action
(`pull an arm')

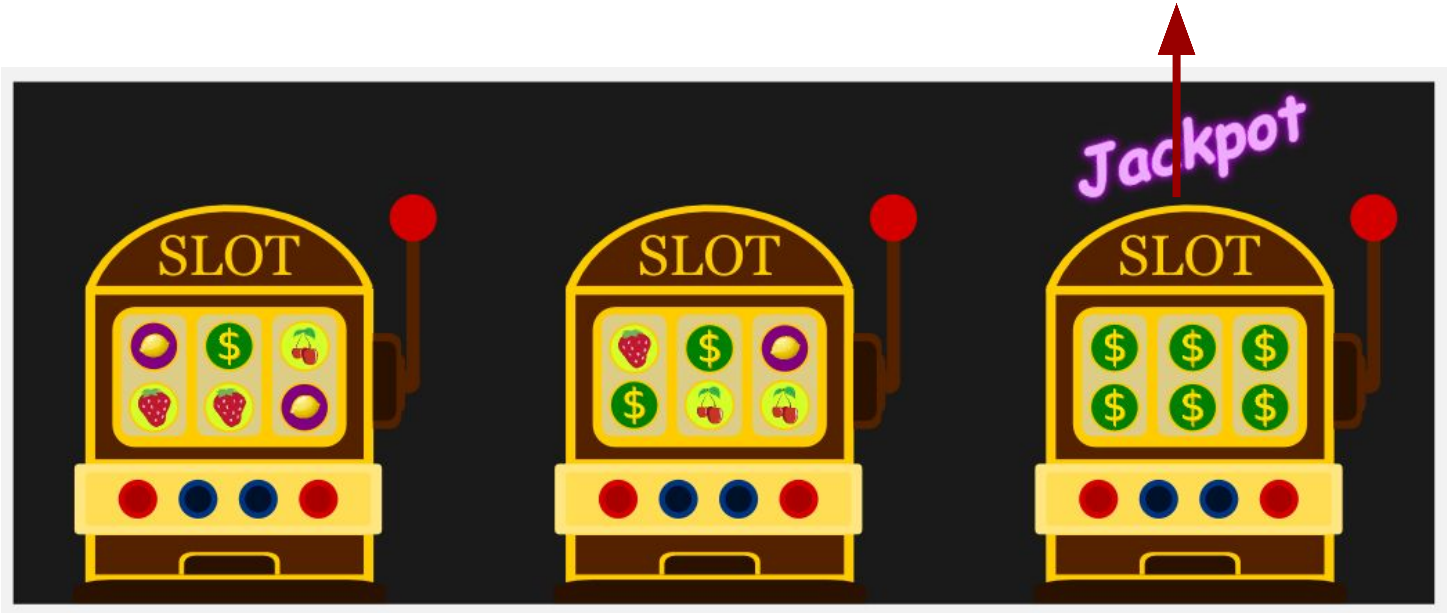
Multi-armed bandit

Reward = -1

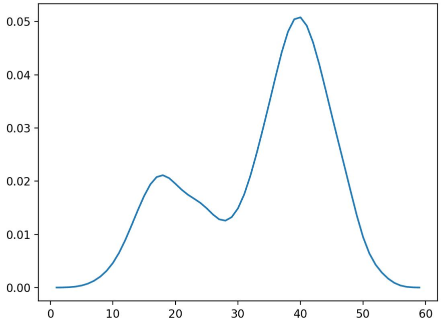


Multi-armed bandit

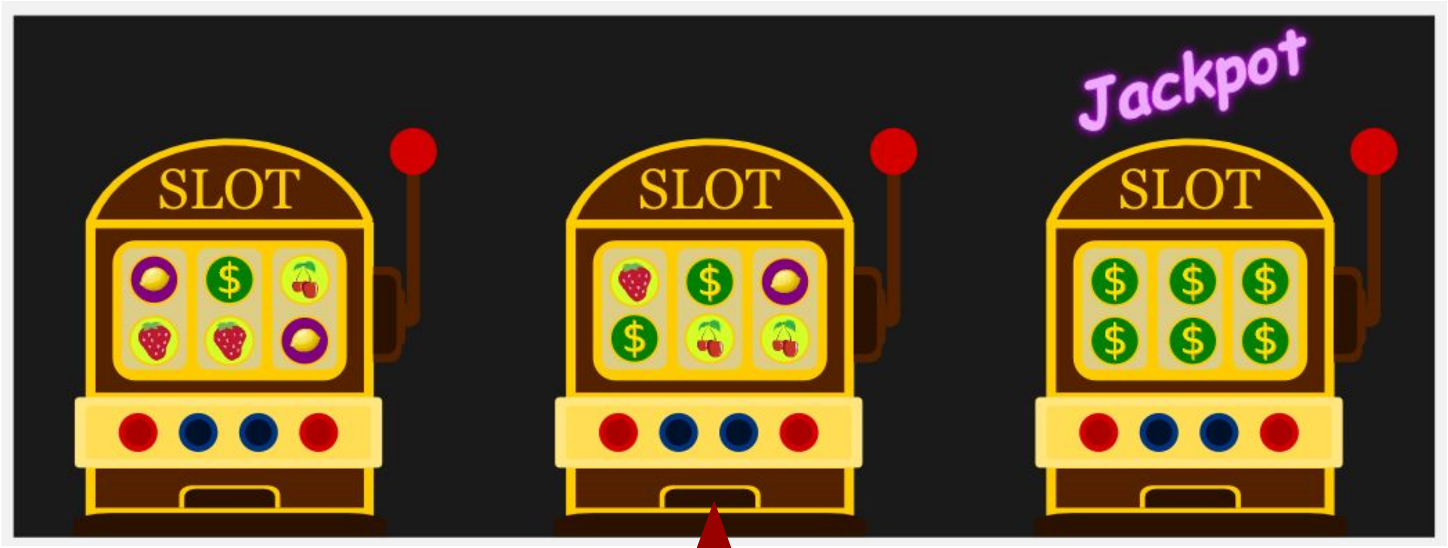
Reward = -1



(noisy sample from unknown reward distribution)



Multi-armed bandit

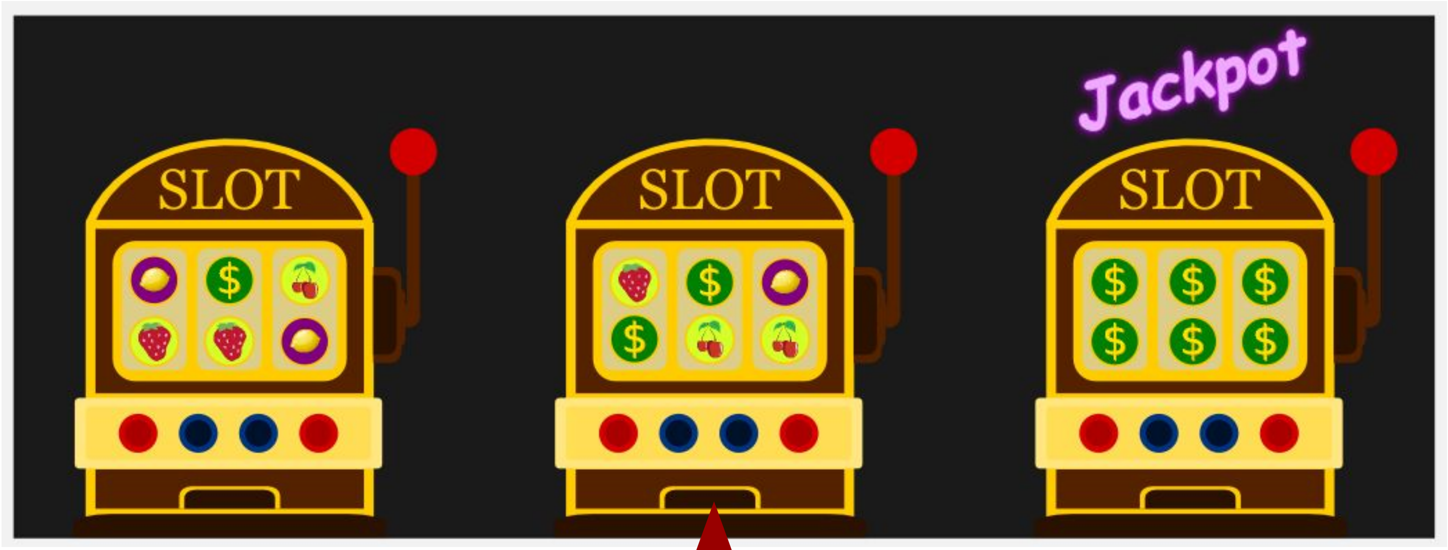


Let's try another one

Multi-armed bandit

Reward = 4

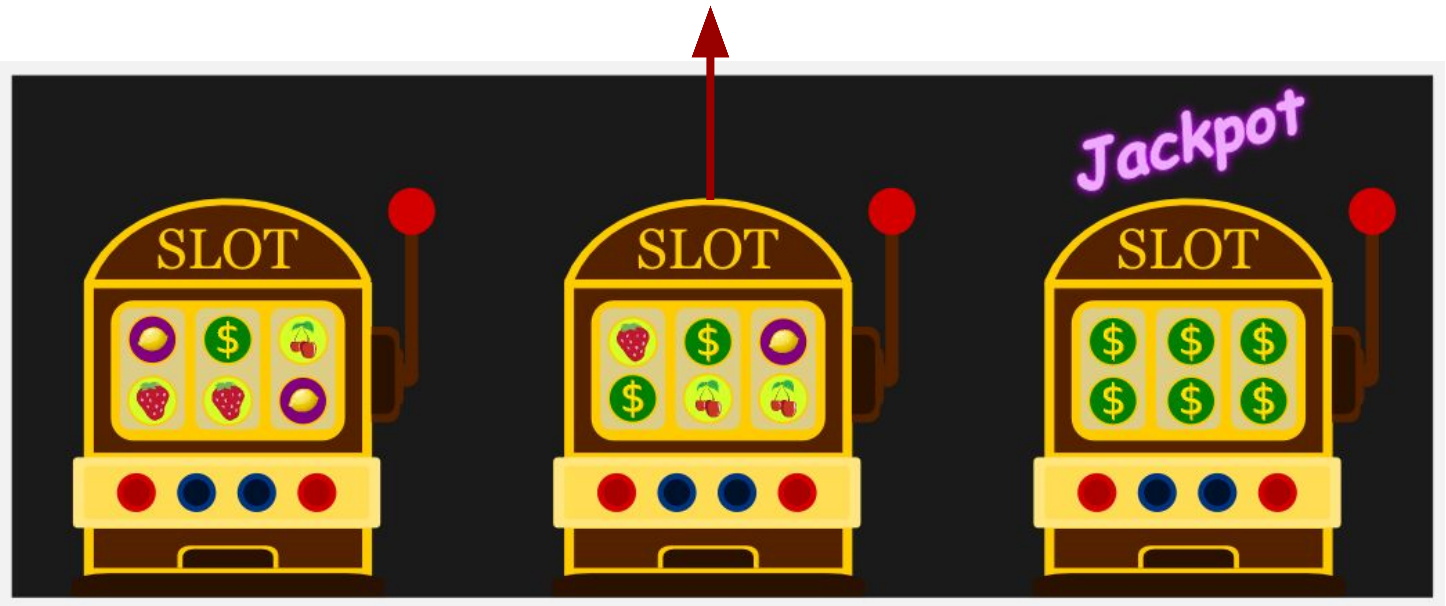
Multi-armed bandit



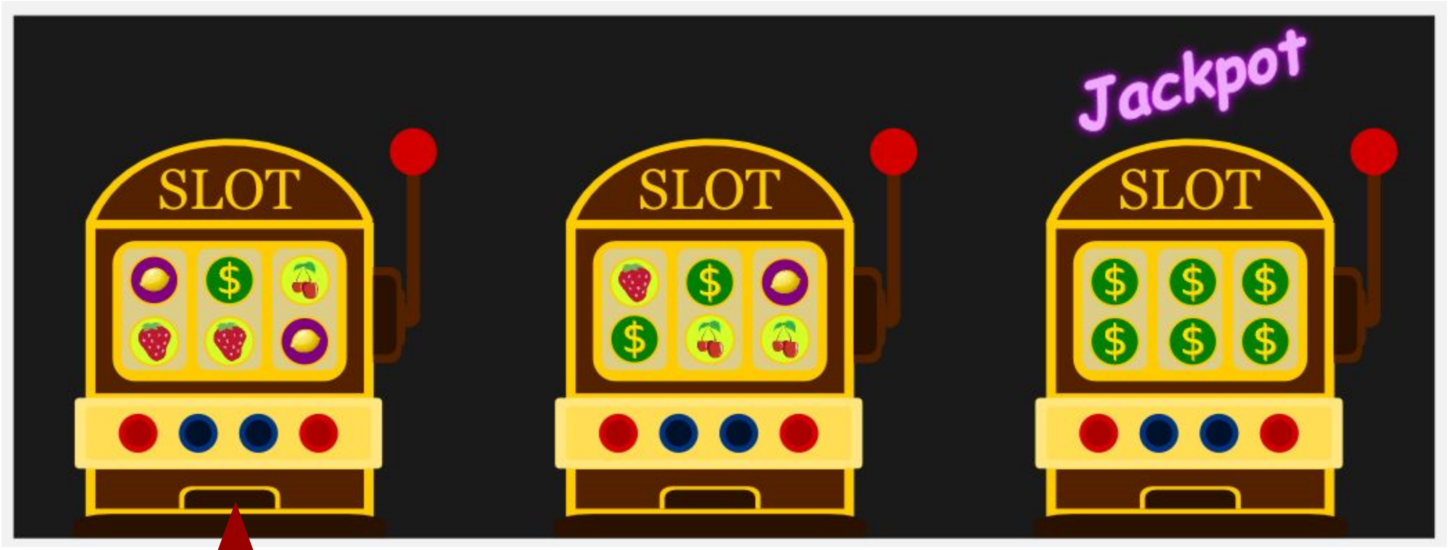
Let's try it again

Multi-armed bandit

Reward = -3

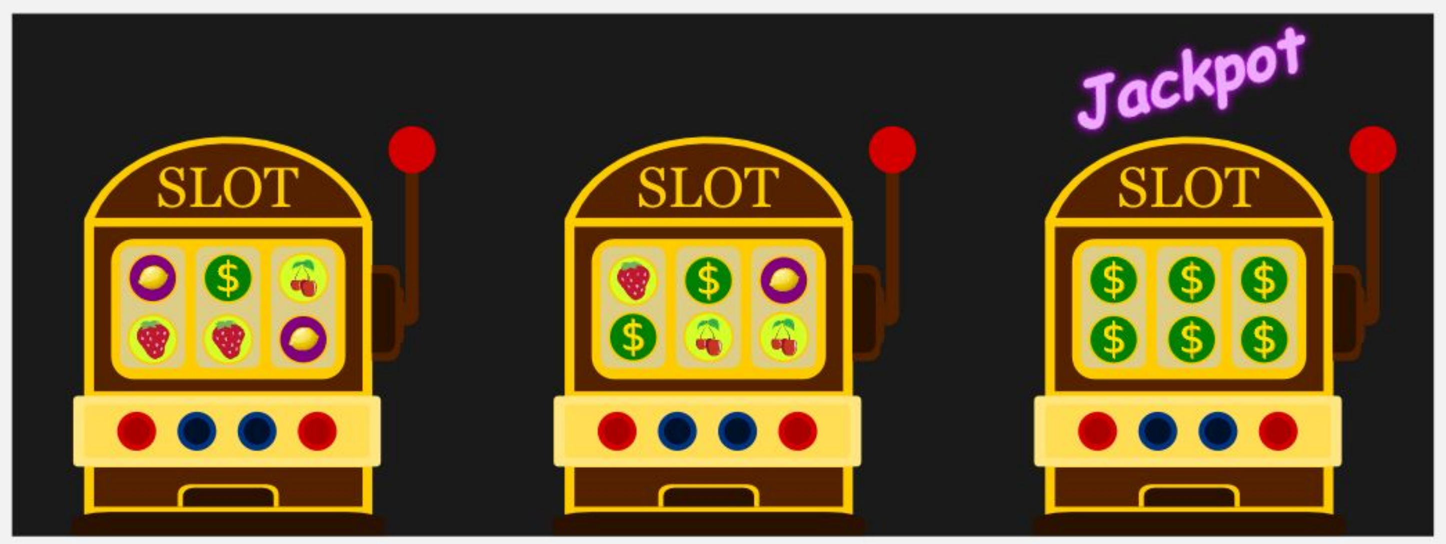


Multi-armed bandit



etc.

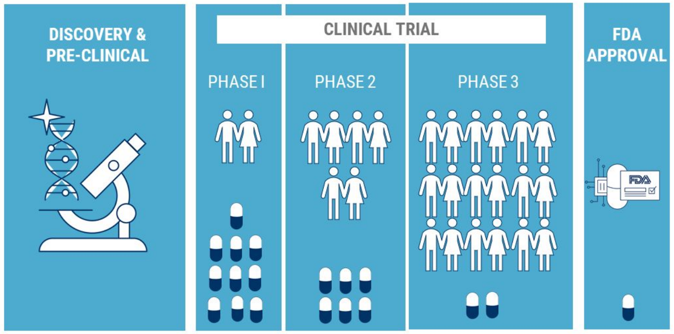
Multi-armed bandit



Our aim: maximize sum of all rewards

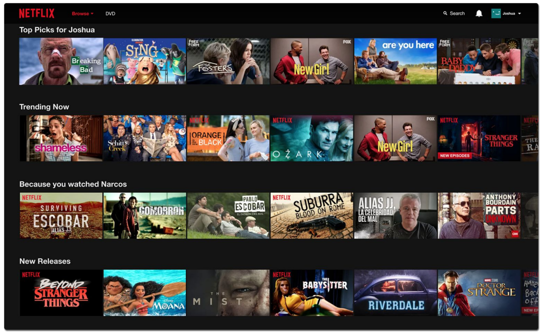
Applications

Applications

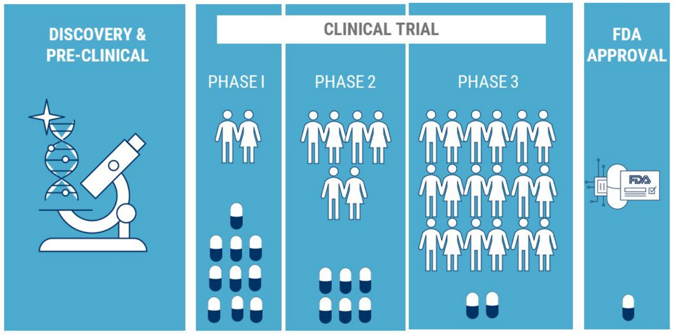


**Medicine:
clinical trials**

Applications

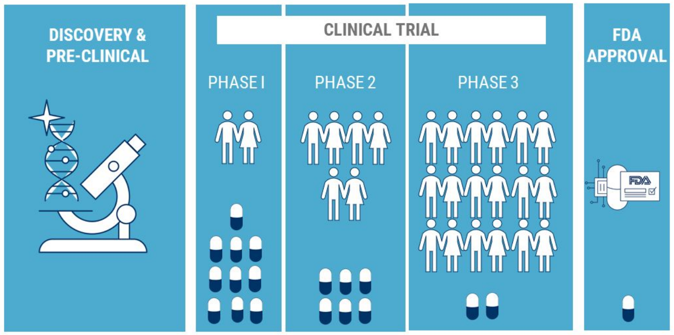


Advertisement: recommender systems

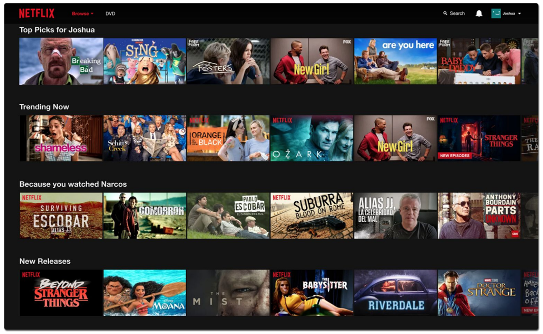


Medicine: clinical trials

Applications



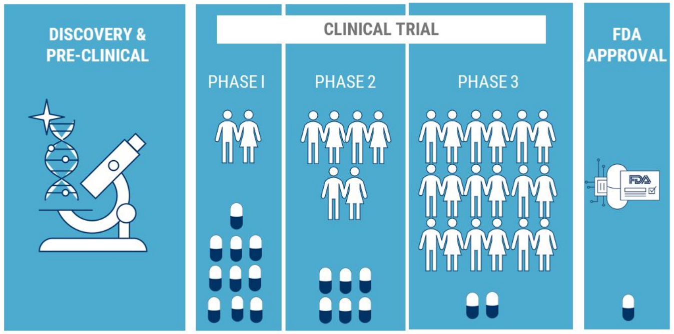
Medicine:
clinical trials



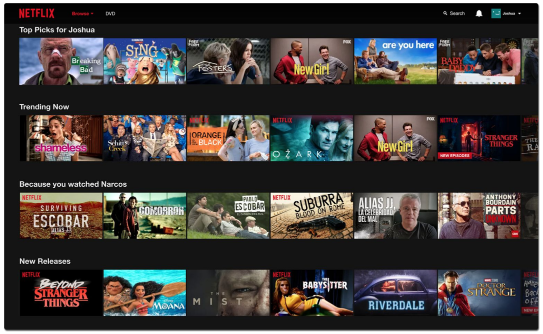
Advertisement:
recommender systems

Finance:
portfolio management

Applications



Medicine: clinical trials



Advertisement: recommender systems

Finance: portfolio management

Also: Important building block for reinforcement learning

Setting

Two important considerations:

Setting

Two important considerations:

- Dataset given or active collection?

Setting

Two important considerations:

- Dataset given or active collection?
- Full feedback (correct prediction) or partial feedback (noisy preference)?

Setting

Two important considerations:

- Dataset given or active collection?
- Full feedback (correct prediction) or partial feedback (noisy preference)?

	Full feedback	Partial feedback
Dataset given		
Actively collect data		

Setting

Two important considerations:

- Dataset given or active collection?
- Full feedback (correct prediction) or partial feedback (noisy preference)?

	Full feedback	Partial feedback
Dataset given	**Supervised learning**	
Actively collect data		

Setting

Two important considerations:

- Dataset given or active collection?
- Full feedback (correct prediction) or partial feedback (noisy preference)?

	Full feedback	Partial feedback
Dataset given	**Supervised learning**	
Actively collect data	**Active learning**	

Setting

Two important considerations:

- Dataset given or active collection?
- Full feedback (correct prediction) or partial feedback (noisy preference)?

	Full feedback	Partial feedback
Dataset given	**Supervised learning**	
Actively collect data	**Active learning**	**Bandits / Reinforcement learning**

Multi-armed bandit: definition

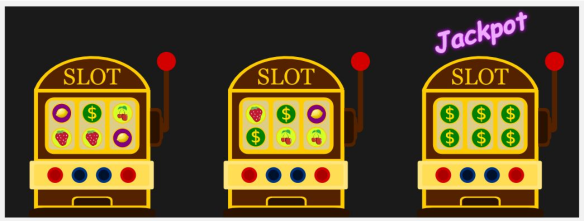
Multi-armed bandit: definition

$$\langle \mathcal{A}, p(r|a) \rangle$$

Multi-armed bandit: definition

$$\langle \mathcal{A}, p(r|a) \rangle$$

Set of actions
e.g.: {1,2,3}

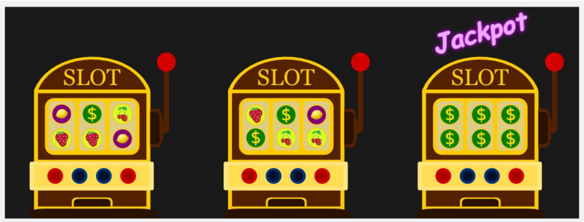


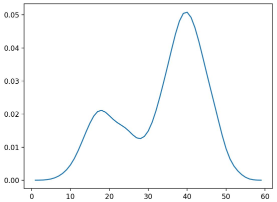
Multi-armed bandit: definition

$$\langle \mathcal{A}, p(r|a) \rangle$$

Set of actions
e.g.: {1,2,3}

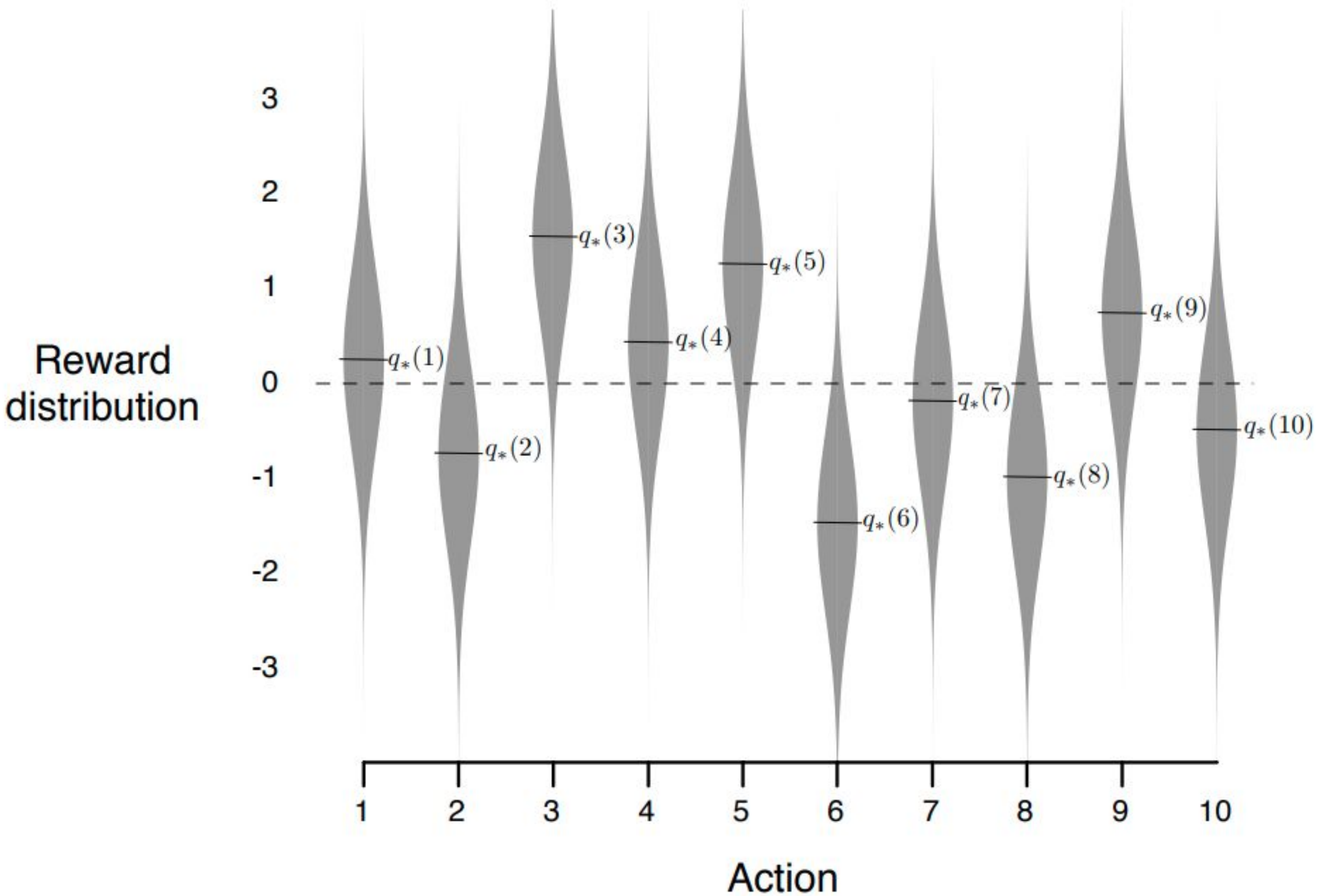
Distribution of rewards
for each action





Multi-armed bandit: example

Multi-armed bandit: example



Action value Q(a)

- Every arm has a *mean pay-off* Q(a):

$$Q(a) = \mathbb{E}_{r \sim p(r|a)}[r]$$

Action value Q(a)

- Every arm has a *mean pay-off* Q(a):

$$Q(a) = \mathbb{E}_{r \sim p(r|a)}[r]$$

"expectation"

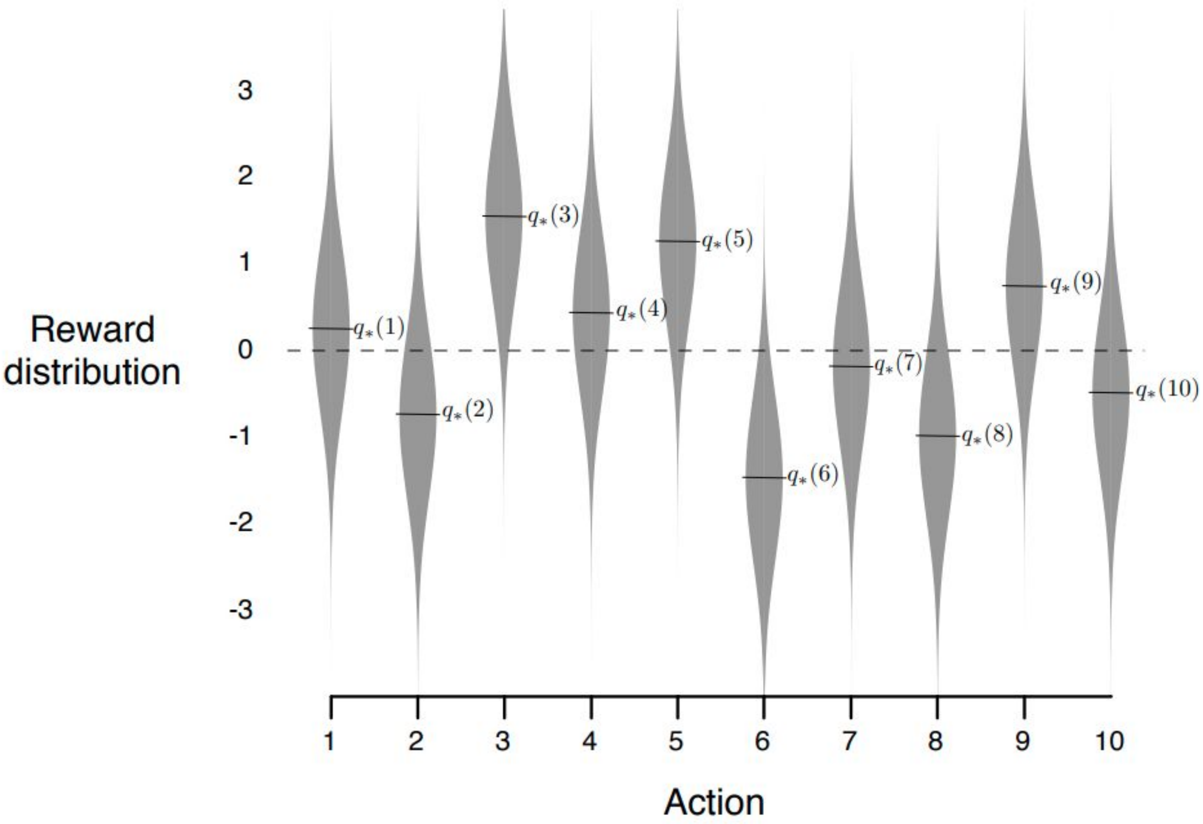
Action value Q(a)

- Every arm has a *mean pay-off* Q(a):

$$Q(a) = \mathbb{E}_{r \sim p(r|a)}[r]$$

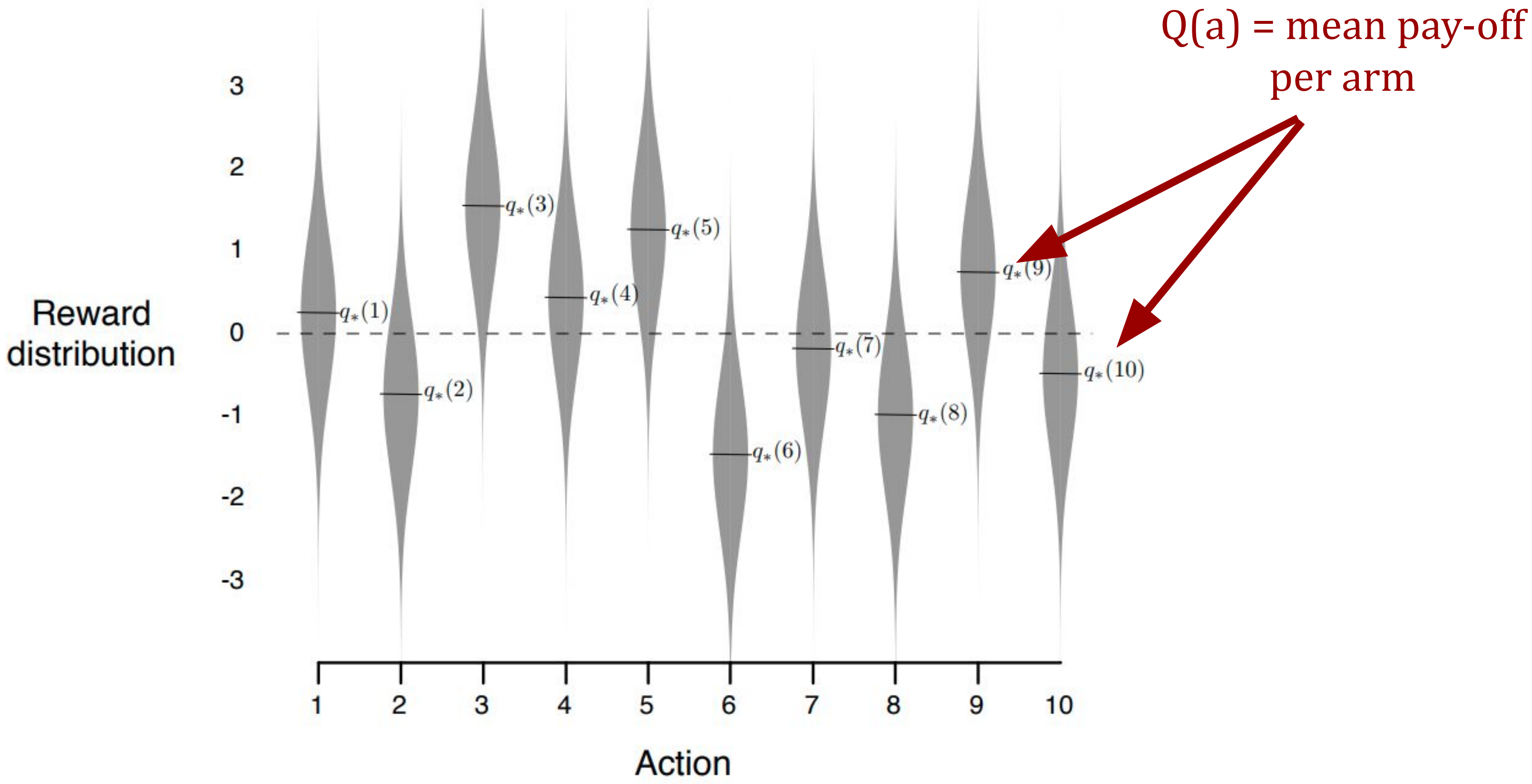
We call Q(a) the **“action value”**

Action value Q(a)



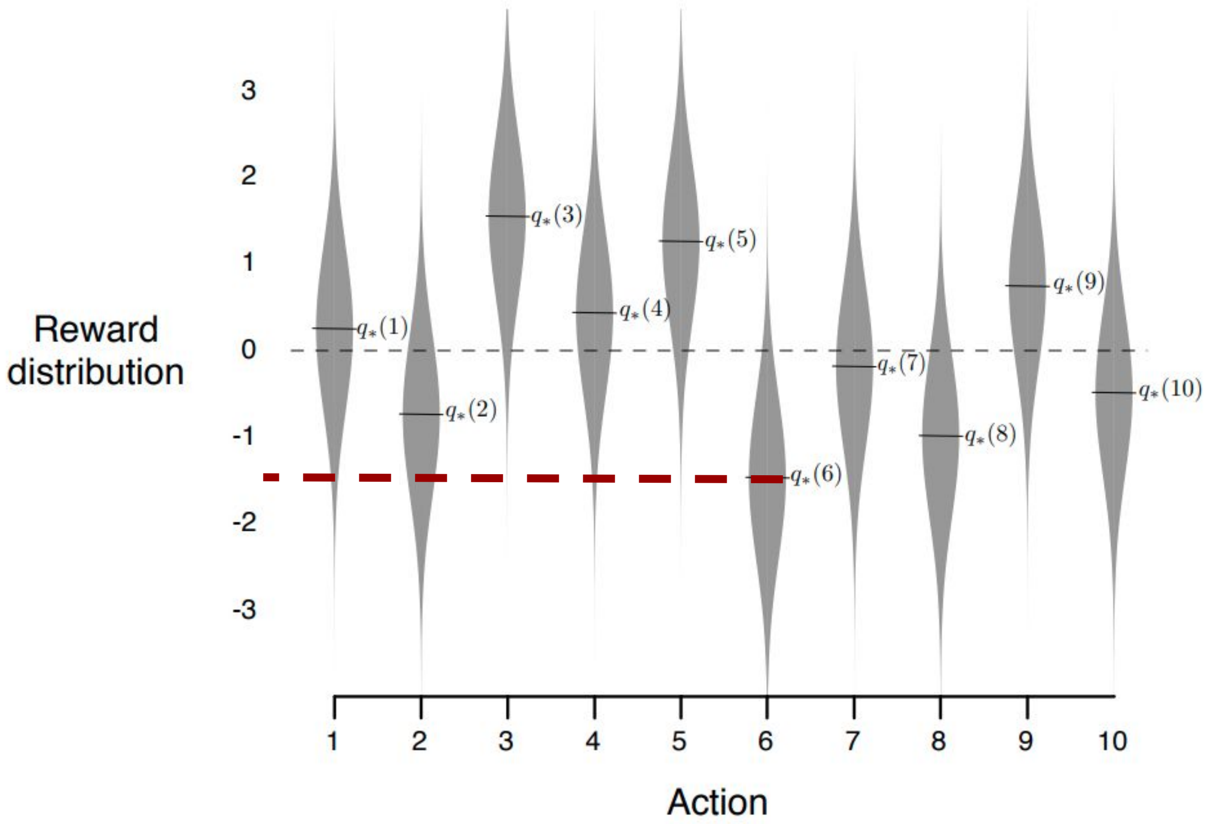
Question: what do you estimate Q(a=6)?

Action value Q(a)



Question: what do you estimate Q(a=6)?

Action value Q(a)



Question: what do you estimate Q(a=6)?

Answer: Q(a=6) ≈ - 1.5

Action selection

Action selection

$$\pi(a)$$

“Policy”

=

Probability distribution over the (discrete) action space

Action selection

$$\pi(a)$$

"Policy"

=

Probability distribution over the (discrete) action space

Example:

$\pi(a=1)$	$\pi(a=2)$	$\pi(a=3)$	$\pi(a=4)$
0.2	0.7	0.0	0.1

Action selection

$$\pi(a)$$

Policy can also be **implicitly** stored

Action selection

$$\pi(a)$$

Policy can also be **implicitly** stored

Insight: close relation between action value Q(a) and policy π(a) (higher action value should get higher policy probability)

Action selection

$$\pi(a)$$

Policy can also be **implicitly** stored

Insight: close relation between action value Q(a) and policy π(a) (higher action value should get higher policy probability)

1. Store the action value estimates:

$Q(a=1)$	$Q(a=2)$	$Q(a=3)$	$Q(a=4)$
1.2	0.3	-2.4	3.5

Action selection

$$\pi(a)$$

Policy can also be **implicitly** stored

Insight: close relation between action value Q(a) and policy π(a) (higher action value should get higher policy probability)

1. Store the action value estimates:

$Q(a=1)$	$Q(a=2)$	$Q(a=3)$	$Q(a=4)$
1.2	0.3	-2.4	3.5

2. Make policy a function of the action values:

$$\pi = f(Q(a))$$

Action selection

$$\pi(a)$$

Policy can also be **implicitly** stored

Insight: close relation between action value Q(a) and policy π(a) (higher action value should get higher policy probability)

1. Store the action value estimates:

$Q(a=1)$	$Q(a=2)$	$Q(a=3)$	$Q(a=4)$
1.2	0.3	-2.4	3.5

2. Make policy a function of the action values:

$$\pi = f(Q(a))$$

Most common: will see different implicit policies (forms of f)!

Objective

What do we actually want to achieve in the bandit setting?

Objective

What do we actually want to achieve in the bandit setting?

Repeatedly choose the right action and get as much reward as possible!

Objective

What do we actually want to achieve in the bandit setting?

$$J_T(\pi) = \mathbb{E}_{a_t \sim \pi(a), r_t \sim p(r|a_t)}[\sum_{t=1}^{T} r_t]$$

Objective

What do we actually want to achieve in the bandit setting?

$$J_T(\pi) = \mathbb{E}_{a_t \sim \pi(a), r_t \sim p(r|a_t)}[\sum_{t=1}^{T} r_t]$$

Objective is a function of our policy

Objective

What do we actually want to achieve in the bandit setting?

$$J_T(\pi) = \mathbb{E}_{a_t \sim \pi(a), r_t \sim p(r|a_t)}[\sum_{t=1}^{T} r_t]$$

Objective is a function of our policy

Average performance over our policy and the stochastic rewards

Objective

What do we actually want to achieve in the bandit setting?

$$J_T(\pi) = \mathbb{E}_{a_t \sim \pi(a), r_t \sim p(r|a_t)}[\sum_{t=1}^{T} r_t]$$

Sum of rewards over T decisions

Objective is a function of our policy

Average performance over our policy and the stochastic rewards

Objective

What do we actually want to achieve in the bandit setting?

$$J_T(\pi) = \mathbb{E}_{a_t \sim \pi(a), r_t \sim p(r|a_t)}[\sum_{t=1}^{T} r_t]$$

But our goal is to find the *best* policy

Objective

What do we actually want to achieve in the bandit setting?

$$J_T(\pi) = \mathbb{E}_{a_t \sim \pi(a), r_t \sim p(r|a_t)}[\sum_{t=1}^{T} r_t]$$

$$\pi^\star = \arg\max_{\pi} J_T(\pi)$$

Objective

What do we actually want to achieve in the bandit setting?

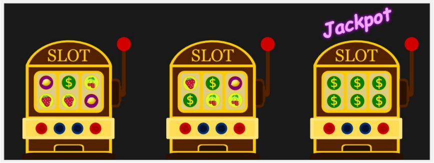
$$J_T(\pi) = \mathbb{E}_{a_t \sim \pi(a), r_t \sim p(r|a_t)}[\sum_{t=1}^{T} r_t]$$

$$\pi^\star = \arg\max_{\pi} J_T(\pi)$$

We want to find/specify π^*, the policy that maximizes our average pay-off!

Summary

- *Bandit definition:* $\langle \mathcal{A}, p(r|a) \rangle$



- *Action value:* $Q(a) = \mathbb{E}_{r\sim p(r|a)}[r]$

$Q(a=1)$	$Q(a=2)$	$Q(a=3)$	$Q(a=4)$
1.2	0.3	-2.4	3.5

- *Policy:* $\pi(a)$

$\pi(a=1)$	$\pi(a=2)$	$\pi(a=3)$	$\pi(a=4)$
0.2	0.7	0.0	0.1

- *Objective:* $J_T(\pi) = \mathbb{E}_{a_t\sim\pi(a), r_t\sim p(r|a_t)}[\sum_{t=1}^{T} r_t]$

Part 2:

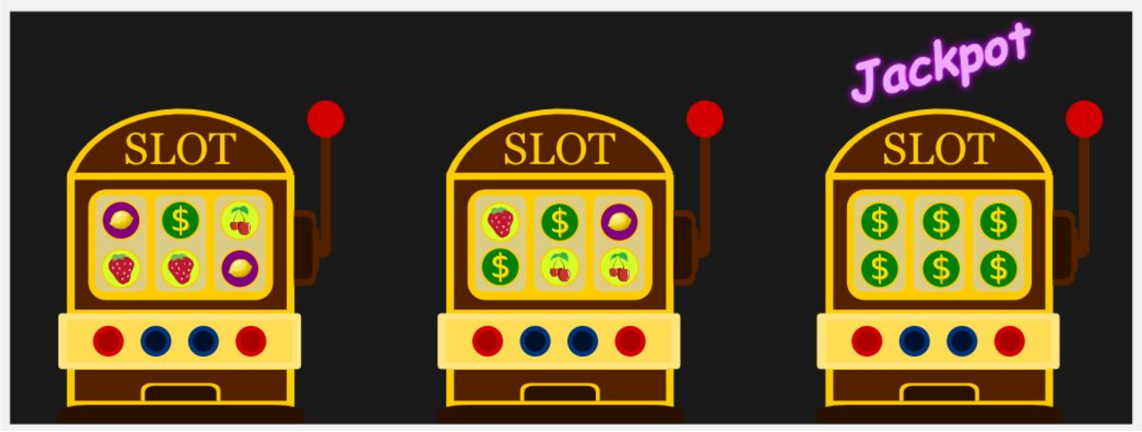
Exploration/Exploitation

Exploration/exploitation

Why is it challenging to find a good policy?

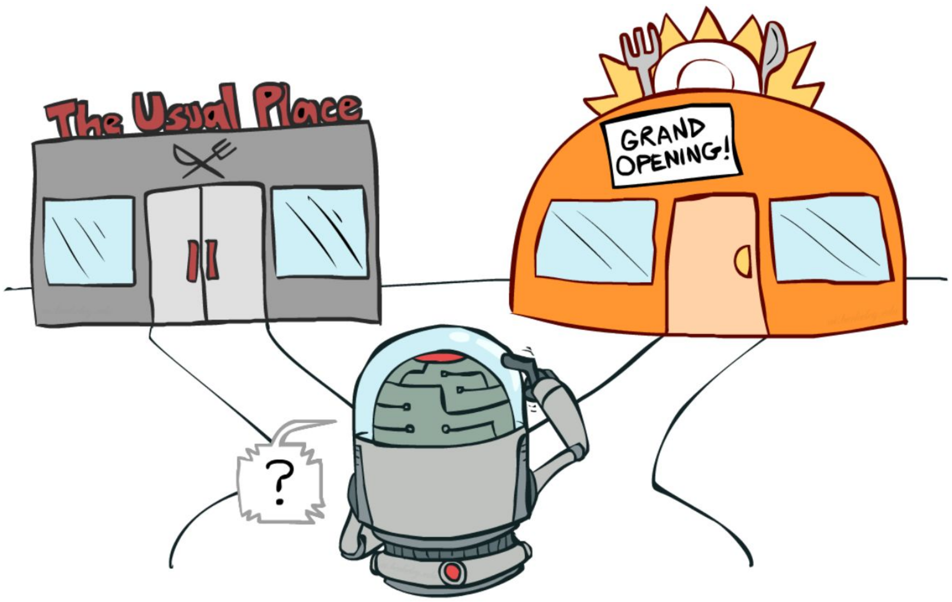
Exploration/exploitation

Why is it challenging to find a good policy?

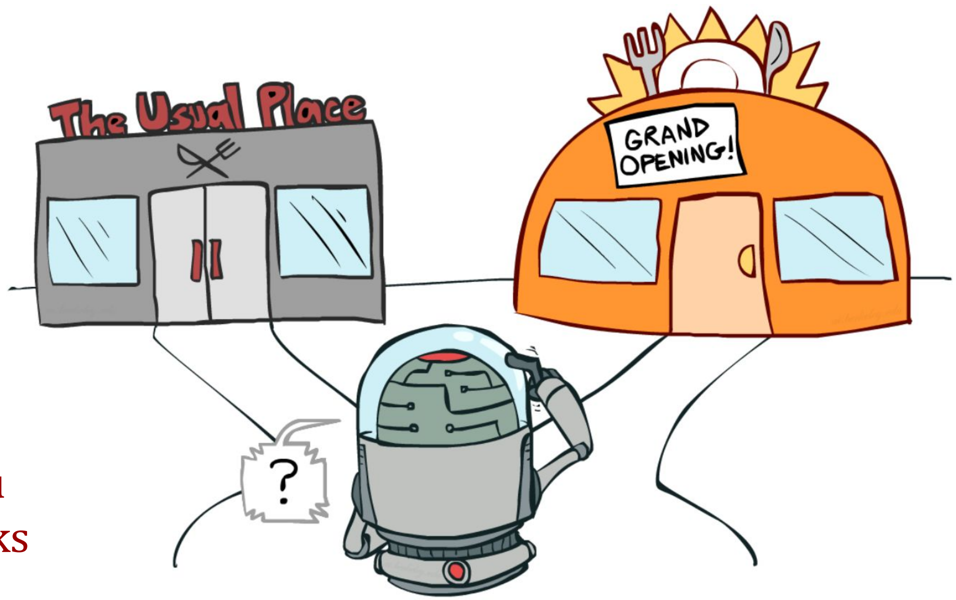


Question: You can play 100 rounds at these slotmachines. How would you act?

Exploration/exploitation

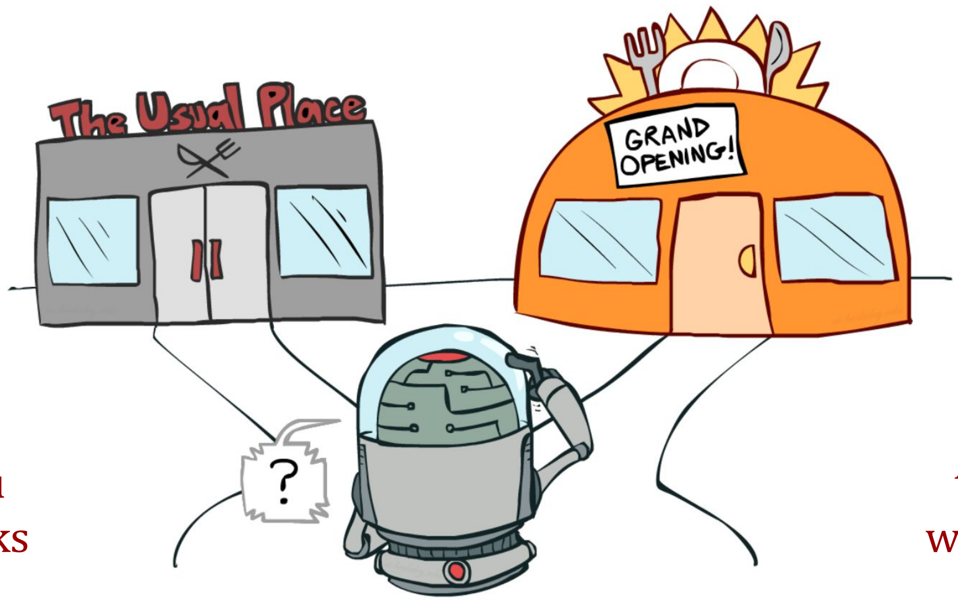


Exploration/exploitation



Exploitation:
do something you
already know works
(reasonably) well

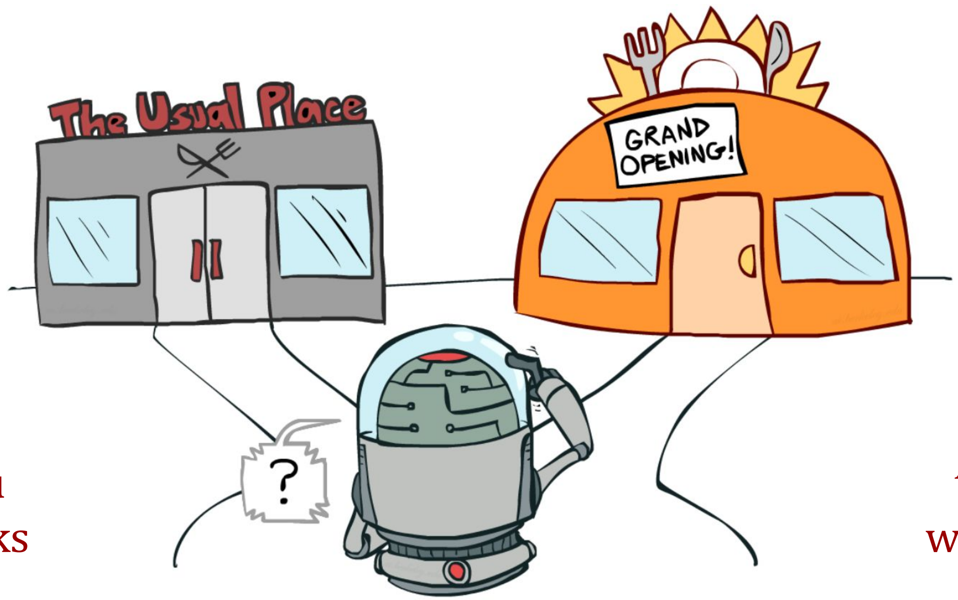
Exploration/exploitation



Exploitation:
do something you already know works (reasonably) well

Exploration:
try something new which may work even better (or worse)

Exploration/exploitation



Exploitation:
do something you already know works (reasonably) well

Exploration:
try something new which may work even better (or worse)

Fundamental trade-off in all decision-making problems (bandits, RL)
(and in life in general)

Bandit algorithm pseudocode

```
Initialization: Initialize policy π(a)
for t = 1...T do
  | a_t ~ π(a)                                  /* Sample from policy */
  | r_t ~ p(r|a_t)                              /* Observe reward */
  | Update π based on (a_t, r_t)
end
```

Bandit algorithm pseudocode

Initialization: Initialize policy $\pi(a)$

for $t = 1 \ldots T$ **do**

  $a_t \sim \pi(a)$ /* Sample from policy */

  $r_t \sim p(r|a_t)$ /* Observe reward */

  Update π based on (a_t, r_t)

end

Bandit algorithm pseudocode

Initialization: Initialize policy $\pi(a)$
for $t = 1...T$ **do**
  $\boxed{a_t \sim \pi(a)}$ `/* Sample from policy */`
  $r_t \sim p(r|a_t)$ `/* Observe reward */`
  Update π based on (a_t, r_t)
end

Exploration/exploitation!

Bandit algorithm pseudocode

Initialization: Initialize policy $\pi(a)$
for $t = 1...T$ **do**

$a_t \sim \pi(a)$ /* Sample from policy */

$\boxed{r_t \sim p(r|a_t)}$ /* Observe reward */

Update π based on (a_t, r_t)

end

Bandit algorithm pseudocode

Initialization: Initialize policy $\pi(a)$
for $t = 1...T$ **do**
  $a_t \sim \pi(a)$ `/* Sample from policy */`
  $r_t \sim p(r|a_t)$ `/* Observe reward */`
  $\boxed{\text{Update } \pi \text{ based on } (a_t, r_t)}$
end

Bandit algorithm pseudocode

Initialization: Initialize policy $\pi(a)$
for $t = 1...T$ **do**
  $\boxed{a_t \sim \pi(a)}$ /* Sample from policy */
  $r_t \sim p(r|a_t)$ /* Observe reward */
  Update π based on (a_t, r_t)
end

Exploration/exploitation!

Bandit algorithm pseudocode

Initialization: Initialize policy $\pi(a)$
for $t = 1...T$ **do**
  $a_t \sim \pi(a)$ `/* Sample from policy */`
  $\boxed{r_t \sim p(r|a_t)}$ `/* Observe reward */`
  Update π based on (a_t, r_t)
end

Bandit algorithm pseudocode

Initialization: Initialize policy $\pi(a)$
for $t = 1...T$ **do**
    $a_t \sim \pi(a)$ /* Sample from policy */
    $r_t \sim p(r|a_t)$ /* Observe reward */
    $\boxed{\text{Update } \pi \text{ based on } (a_t, r_t)}$
end

etc.

Part 3:

Updating a mean

Estimate a mean

Many algorithms rely on our ability to estimate the *mean* reward of an action:

Estimate a mean

Many algorithms rely on our ability to estimate the *mean* reward of an action:

$$Q_n = \frac{r_1 + r_2 + ... + r_n}{n} = \frac{1}{n}\sum_{i=1}^{n} r_i$$

Estimate a mean

Many algorithms rely on our ability to estimate the *mean* reward of an action:

$$Q_n = \frac{r_1 + r_2 + ... + r_n}{n} = \frac{1}{n}\sum_{i=1}^{n} r_i$$

Q: For arm a you observe r_1=4, r_2=7, r_3=1. What is Q(a)?

Estimate a mean

Many algorithms rely on our ability to estimate the *mean* reward of an action:

$$Q_n = \frac{r_1 + r_2 + ... + r_n}{n} = \frac{1}{n}\sum_{i=1}^{n} r_i$$

Q: For arm a you observe r_1=4, r_2=7, r_3=1. What is Q(a)?

A: (4 + 7 + 1) / 3 = 4

Estimate a mean

Many algorithms rely on our ability to estimate the *mean* reward of an action:

$$Q_n = \frac{r_1 + r_2 + ... + r_n}{n} = \frac{1}{n}\sum_{i=1}^{n} r_i$$

However, now next r_{n+1} comes in, and *how do we update the current mean* $Q_{n?}$*?*

1) Incremental update
2) Learning update

Incremental mean update

Can we write Q_n as a function of the ***previous mean*** *Q_{n-1} and the* ***new reward*** *r_n?*

Incremental mean update

$$Q_n = \frac{1}{n}\sum_{i=1}^{n} r_i$$

Incremental mean update

$$
\begin{aligned}
Q_n &= \frac{1}{n}\sum_{i=1}^{n} r_i \\
&= \frac{1}{n}[r_n + \sum_{i=1}^{n-1} r_i]
\end{aligned}
$$

Incremental mean update

$$
\begin{aligned}
Q_n &= \frac{1}{n}\sum_{i=1}^{n} r_i \\
&= \frac{1}{n}[r_n + \sum_{i=1}^{n-1} r_i] \\
&= \frac{1}{n}[r_n + (n-1)\frac{1}{(n-1)}\sum_{i=1}^{n-1} r_i]
\end{aligned}
$$

Incremental mean update

$$
\begin{aligned}
Q_n &= \frac{1}{n}\sum_{i=1}^{n} r_i \\
&= \frac{1}{n}[r_n + \sum_{i=1}^{n-1} r_i] \\
&= \frac{1}{n}[r_n + (n-1)\frac{1}{(n-1)}\sum_{i=1}^{n-1} r_i] \\
&= \frac{1}{n}[r_n + (n-1)Q_{n-1}]
\end{aligned}
$$

Incremental mean update

$$
\begin{aligned}
Q_n &= \frac{1}{n}\sum_{i=1}^{n} r_i \\
&= \frac{1}{n}[r_n + \sum_{i=1}^{n-1} r_i] \\
&= \frac{1}{n}[r_n + (n-1)\frac{1}{(n-1)}\sum_{i=1}^{n-1} r_i] \\
&= \frac{1}{n}[r_n + (n-1)Q_{n-1}] \\
&= \frac{1}{n}[r_n + n \cdot Q_{n-1} - Q_{n-1}]
\end{aligned}
$$

Incremental mean update

$$
\begin{aligned}
Q_n &= \frac{1}{n}\sum_{i=1}^{n} r_i \\
&= \frac{1}{n}[r_n + \sum_{i=1}^{n-1} r_i] \\
&= \frac{1}{n}[r_n + (n-1)\frac{1}{(n-1)}\sum_{i=1}^{n-1} r_i] \\
&= \frac{1}{n}[r_n + (n-1)Q_{n-1}] \\
&= \frac{1}{n}[r_n + n \cdot Q_{n-1} - Q_{n-1}] \\
Q_n &= Q_{n-1} + \frac{1}{n}[r_n - Q_{n-1}]
\end{aligned}
$$

Incremental mean update

$$
\begin{aligned}
Q_n &= \frac{1}{n}\sum_{i=1}^{n} r_i \\
&= \frac{1}{n}[r_n + \sum_{i=1}^{n-1} r_i] \\
&= \frac{1}{n}[r_n + (n-1)\frac{1}{(n-1)}\sum_{i=1}^{n-1} r_i] \\
&= \frac{1}{n}[r_n + (n-1)Q_{n-1}] \\
&= \frac{1}{n}[r_n + n \cdot Q_{n-1} - Q_{n-1}] \\
Q_n &= Q_{n-1} + \frac{1}{n}[r_n - Q_{n-1}]
\end{aligned}
$$

Not for lecture: check this derivation in the book/lecture slides!

Incremental mean update

$$\boxed{Q_n = Q_{n-1} + \frac{1}{n}[r_n - Q_{n-1}]}$$

Incremental update rule for the mean

Incremental mean update

Question:

- For action 1 we took 3 samples so far, with a mean reward of 4.0
- We take a new sample, and observe a reward of 6.0
- Compute the new Q(a).

$$\boxed{Q_n = Q_{n-1} + \frac{1}{n}[r_n - Q_{n-1}]}$$

Incremental update rule for the mean

Incremental mean update

Question:

- For action 1 we took 3 samples so far, with a mean reward of 4.0
- We take a new sample, and observe a reward of 6.0
- Compute the new Q(a).

Answer: 4.0 + ¼ * [6.0 - 4.0] = = 4.5

$$\boxed{Q_n = Q_{n-1} + \frac{1}{n}[r_n - Q_{n-1}]}$$

Incremental update rule for the mean

Learning mean update

‘Simply move the new mean a bit in the direction of the last observed reward’

Learning mean update

‘Simply move the new mean a bit in the direction of the last observed reward’

$$Q_n \leftarrow Q_{n-1} + \alpha \Big[r_n - Q_{n-1} \Big]$$

Learning mean update

‘Simply move the new mean a bit in the direction of the last observed reward’

$$Q_n \leftarrow Q_{n-1} + \alpha \Big[r_n - Q_{n-1} \Big]$$

Learning rate α

Learning mean update

‘Simply move the new mean a bit in the direction of the last observed reward’

$$Q_n \leftarrow Q_{n-1} + \alpha \Big[r_n - Q_{n-1} \Big]$$

Question:

- For action 1 we took 3 samples so far, with a mean reward of 4.0
- We take a new sample, and observe a reward of 6.0
- Compute the new Q(a) for learning rate = 0.2

Learning mean update

'Simply move the new mean a bit in the direction of the last observed reward'

$$Q_n \leftarrow Q_{n-1} + \alpha\Big[r_n - Q_{n-1}\Big]$$

Question:

- For action 1 we took 3 samples so far, with a mean reward of 4.0
- We take a new sample, and observe a reward of 6.0
- Compute the new Q(a) for learning rate = 0.2

Answer: 4.0 + 0.2 * [6.0 - 4.0] = = 4.4

Summary: update of mean

- *Incremental update:*

- *Learning update:*

Summary: update of mean

- *Incremental update:*

$$Q_n = Q_{n-1} + \frac{1}{n}\Big[r_n - Q_{n-1}\Big]$$

- *Learning update:*

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

Summary: update of mean

- *Incremental update:*

$$Q_n = Q_{n-1} + \boxed{\frac{1}{n}} \left[r_n - Q_{n-1} \right]$$

- *Learning update:*

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

Summary: update of mean

- *Incremental update:*

$$Q_n = Q_{n-1} + \boxed{\frac{1}{n}}\left[r_n - Q_{n-1}\right]$$

Equally weights each sample

- *Learning update:*

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

More weight on recent samples

Break

Part 4:

Bandit algorithms

Bandit pseudocode

Bandit pseudocode:

Initialization: Initialize policy $\pi(a)$

for $t = 1...T$ **do**

  $a_t \sim \pi(a)$ /* Sample from policy */

  $r_t \sim p(r|a_t)$ /* Observe reward */

  Update π based on (a_t, r_t)

end

Bandit pseudocode

Bandit pseudocode:

Initialization: $\boxed{\text{Initialize}}$ policy $\pi(a)$
for $t = 1 \ldots T$ **do**
  $a_t \sim \boxed{\pi(a)}$ /* Sample from policy */
  $r_t \sim p(r \mid a_t)$ /* Observe reward */
  $\boxed{\text{Update } \pi}$ based on (a_t, r_t)
end

Bandit pseudocode

Bandit pseudocode:

Initialization: $\boxed{\text{Initialize}}$ policy $\pi(a)$
for $t = 1...T$ **do**
  $a_t \sim \boxed{\pi(a)}$ /* Sample from policy */
  $r_t \sim p(r|a_t)$ /* Observe reward */
  $\boxed{\text{Update } \pi}$ based on (a_t, r_t)
end

Three main things we need to decide on:

1. Initialization
2. Action selection (exploration/exploitation)
3. Updating

Bandit pseudocode

Bandit pseudocode:

Initialization: Initialize policy $\pi(a)$
for $t = 1...T$ **do**
  $a_t \sim \pi(a)$ `/* Sample from policy */`
  $r_t \sim p(r|a_t)$ `/* Observe reward */`
  Update π based on (a_t, r_t)
end

Three main things we need to decide on:

1. Initialization
2. Action selection (exploration/exploitation)
3. **Updating**

Before the break we already discussed two ways to update a mean estimate

Bandit pseudocode

Bandit pseudocode:

Initialization: $\boxed{\text{Initialize}}$ policy $\pi(a)$
for $t = 1...T$ **do**
  $a_t \sim \boxed{\pi(a)}$ /* Sample from policy */
  $r_t \sim p(r|a_t)$ /* Observe reward */
  $\boxed{\text{Update } \pi}$ based on (a_t, r_t)
end

Three main things we need to decide on:

1. Initialization
2. **Action selection (exploration/exploitation)**
3. Updating

Action selection: exploitation

Exploitation:

Action selection: exploitation

Exploitation:

select action with current highest mean estimate

$$\pi_{\text{greedy}}(a) = f(Q) = \begin{cases} 1, & \text{if } a = \arg\max_{b \in \mathcal{A}} Q(b) \\ 0, & \text{otherwise} \end{cases}$$

Action selection: exploitation

Exploitation:

select action with current highest mean estimate

$$\pi_{\text{greedy}}(a) = f(Q) = \begin{cases} 1, & \text{if } a = \arg\max_{b \in \mathcal{A}} Q(b) \\ 0, & \text{otherwise} \end{cases}$$

Example:

$Q(a=1)$	$Q(a=2)$	$Q(a=3)$	$Q(a=4)$
1.2	0.3	-2.4	3.5

Action selection: exploitation

Exploitation:

select action with current highest mean estimate

$$\pi_{\text{greedy}}(a) = f(Q) = \begin{cases} 1, & \text{if } a = \arg\max_{b \in \mathcal{A}} Q(b) \\ 0, & \text{otherwise} \end{cases}$$

Example:

$Q(a=1)$	$Q(a=2)$	$Q(a=3)$	$Q(a=4)$
1.2	0.3	-2.4	3.5

Question: what probability will each action get?

Action selection: exploitation

Exploitation:

select action with current highest mean estimate

$$\pi_{\text{greedy}}(a) = f(Q) = \begin{cases} 1, & \text{if } a = \arg\max_{b \in \mathcal{A}} Q(b) \\ 0, & \text{otherwise} \end{cases}$$

Example:

$Q(a=1)$	$Q(a=2)$	$Q(a=3)$	$Q(a=4)$
1.2	0.3	-2.4	3.5

Question: what probability will each action get?

Answer:

$\pi(a=1) = 0.0$

$\pi(a=2) = 0.0$

$\pi(a=3) = 0.0$

$\pi(a=4) = 1.0$

Action selection: exploitation

Exploitation:

select action with current highest mean estimate

$$\pi_{\text{greedy}}(a) = f(Q) = \begin{cases} 1, & \text{if } a = \arg\max_{b \in \mathcal{A}} Q(b) \\ 0, & \text{otherwise} \end{cases}$$

Example:

$Q(a=1)$	$Q(a=2)$	$Q(a=3)$	$\boxed{Q(a=4)}$
1.2	0.3	-2.4	$\boxed{3.5}$

"Greedy policy"

Question: what probability will each action get?

Action selection: exploration

We need to introduce exploration

Action selection: exploration

We need to introduce exploration

Discuss three possible approaches:

1. Random perturbation (ε-greedy)
2. Optimistic initialization (oi)
3. Uncertainty-based (ucb)

Action selection: exploration

We need to introduce exploration

Discuss three possible approaches:

1. **Random perturbation** **(ε-greedy)**
2. Optimistic initialization (oi)
3. Uncertainty-based (ucb)

ε-greedy

"Act greedily, but with (small) probability ε, sample a random other action"

ε-greedy

"Act greedily, but with (small) probability ε, sample a random other action"

$$\pi_{\epsilon-\text{greedy}}(a) = f(Q,\epsilon) = \begin{cases} 1-\epsilon, & \text{if } a = \arg\max_{b\in\mathcal{A}} Q(b) \\ \frac{\epsilon}{(|\mathcal{A}|-1)}, & \text{otherwise} \end{cases}$$

ε-greedy

"Act greedily, but with (small) probability ε, sample a random other action"

$$\pi_{\epsilon-\mathrm{greedy}}(a) = f(Q, \epsilon) = \begin{cases} 1-\epsilon, & \text{if } a = \arg\max_{b \in \mathcal{A}} Q(b) \\ \frac{\epsilon}{(|\mathcal{A}|-1)}, & \text{otherwise} \end{cases}$$

ε **= exploration parameter**

scale the amount of exploration

ε-greedy

"Act greedily, but with (small) probability ε, sample a random other action"

$$\pi_{\epsilon-\text{greedy}}(a) = f(Q,\epsilon) = \begin{cases} 1-\epsilon, & \text{if } a = \arg\max_{b\in\mathcal{A}} Q(b) \\ \frac{\epsilon}{(|\mathcal{A}|-1)}, & \text{otherwise} \end{cases}$$

Example

$Q(a=1)$	$Q(a=2)$	$Q(a=3)$	$Q(a=4)$
1.2	0.3	-2.4	3.5

ε-greedy

"Act greedily, but with (small) probability ε, sample a random other action"

$$\pi_{\epsilon-\text{greedy}}(a) = f(Q, \epsilon) = \begin{cases} 1-\epsilon, & \text{if } a = \arg\max_{b \in \mathcal{A}} Q(b) \\ \frac{\epsilon}{(|\mathcal{A}|-1)}, & \text{otherwise} \end{cases}$$

Example

$Q(a=1)$	$Q(a=2)$	$Q(a=3)$	$Q(a=4)$
1.2	0.3	-2.4	3.5

Question: Assume *ε =0.15. What probability of selection will each action get?*

ε-greedy

"Act greedily, but with (small) probability ε, sample a random other action"

$$\pi_{\epsilon-\text{greedy}}(a) = f(Q, \epsilon) = \begin{cases} 1-\epsilon, & \text{if } a = \arg\max_{b \in \mathcal{A}} Q(b) \\ \frac{\epsilon}{(|\mathcal{A}|-1)}, & \text{otherwise} \end{cases}$$

Example

$Q(a=1)$	$Q(a=2)$	$Q(a=3)$	$Q(a=4)$
1.2	0.3	-2.4	3.5

Answer:

$\pi(a=1) = 0.05$

$\pi(a=2) = 0.05$

$\pi(a=3) = 0.05$

$\pi(a=4) = 0.85$

Question: Assume *ε =0.15. What probability of selection will each action get?*

ε-greedy pseudocode

Algorithm 2: ϵ-greedy bandit algorithm.

Input: Exploration parameter $\epsilon \in [0, 1]$, maximum number of timesteps T.

Initialization: Initialize $Q(a) = 0$, $n(a) = 0 \ \forall a \in \mathcal{A}$

for $t = 1 \ldots T$ **do**

$$a_t = \begin{cases} \arg\max_{a \in \mathcal{A}} Q(a) & \text{with } p = 1 - \epsilon \\ \text{random}, & \text{otherwise} \end{cases} \qquad \texttt{/* } \epsilon\texttt{-greedy action */}$$

$$r_t \sim p(r|a_t) \qquad \texttt{/* Sample reward */}$$

$$n(a_t) \leftarrow n(a_t) + 1 \qquad \texttt{/* Update count */}$$

$$Q(a_t) \leftarrow Q(a_t) + \frac{1}{n(a_t)}\Big[r_t - Q(a_t)\Big] \qquad \texttt{/* Incr. update mean */}$$

end

Algorithm now requires an input ε

ε-greedy pseudocode

Algorithm 2: ϵ-greedy bandit algorithm.

Input: Exploration parameter $\epsilon \in [0, 1]$, maximum number of timesteps T.

Initialization: $\boxed{\text{Initialize } Q(a) = 0,\ n(a) = 0\ \forall a \in \mathcal{A}}$

for $t = 1 \ldots T$ **do**

$$a_t = \begin{cases} \arg\max_{a \in \mathcal{A}} Q(a) & \text{with } p = 1 - \epsilon \\ \text{random,} & \text{otherwise} \end{cases}$$ `/* ε-greedy action */`

$r_t \sim p(r|a_t)$ `/* Sample reward */`

$n(a_t) \leftarrow n(a_t) + 1$ `/* Update count */`

$Q(a_t) \leftarrow Q(a_t) + \frac{1}{n(a_t)} \left[r_t - Q(a_t) \right]$ `/* Incr. update mean */`

end

Initialize means and counts to 0

ε-greedy pseudocode

Algorithm 2: ϵ-greedy bandit algorithm.

Input: Exploration parameter $\epsilon \in [0, 1]$, maximum number of timesteps T.

Initialization: Initialize $Q(a) = 0$, $n(a) = 0 \ \forall a \in \mathcal{A}$

for $t = 1...T$ **do**

$$\boxed{a_t = \begin{cases} \arg\max_{a \in \mathcal{A}} Q(a) & \text{with } p = 1 - \epsilon \\ \text{random,} & \text{otherwise} \end{cases} \qquad \texttt{/* } \epsilon\texttt{-greedy action */}}$$

$r_t \sim p(r|a_t)$ `/* Sample reward */`

$n(a_t) \leftarrow n(a_t) + 1$ `/* Update count */`

$Q(a_t) \leftarrow Q(a_t) + \frac{1}{n(a_t)}\left[r_t - Q(a_t)\right]$ `/* Incr. update mean */`

end

Select *ε-greedy action*

ε-greedy pseudocode

Algorithm 2: ϵ-greedy bandit algorithm.

Input: Exploration parameter $\epsilon \in [0, 1]$, maximum number of timesteps T.

Initialization: Initialize $Q(a) = 0$, $n(a) = 0 \; \forall a \in \mathcal{A}$

for $t = 1 \ldots T$ **do**

$$a_t = \begin{cases} \arg\max_{a \in \mathcal{A}} Q(a) & \text{with } p = 1 - \epsilon \\ \text{random,} & \text{otherwise} \end{cases}$$ `/* ϵ-greedy action */`

$r_t \sim p(r|a_t)$ `/* Sample reward */`

$n(a_t) \leftarrow n(a_t) + 1$ `/* Update count */`

$\boxed{Q(a_t) \leftarrow Q(a_t) + \frac{1}{n(a_t)}\left[r_t - Q(a_t)\right]}$ `/* Incr. update mean */`

end

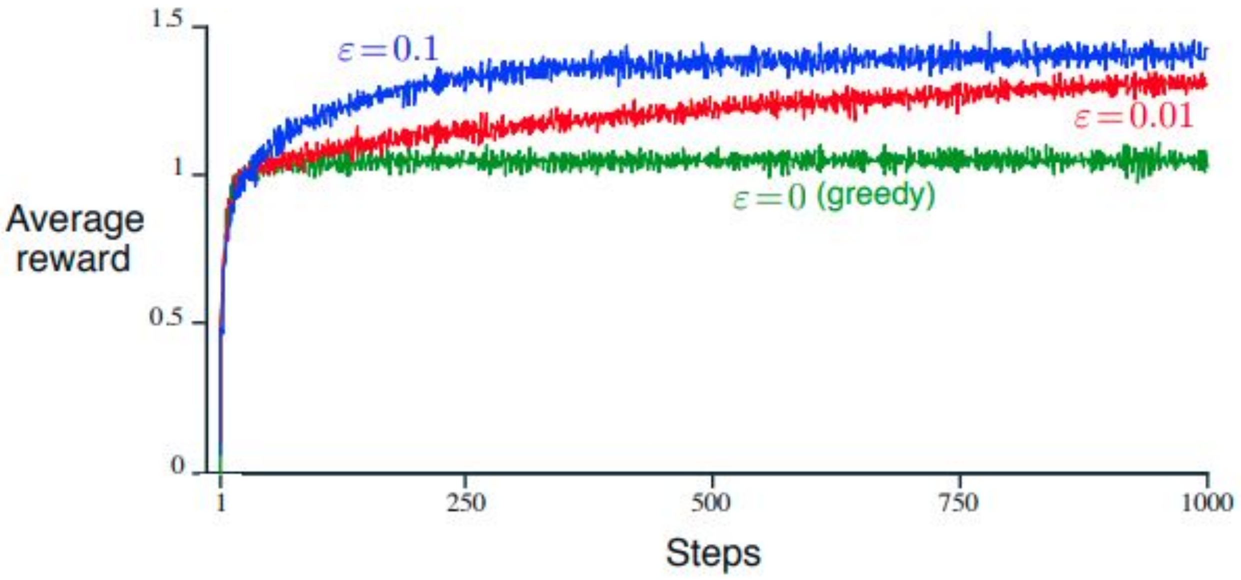
Incremental update to track the means

ε-greedy performance

Exploration parameters should usually be neither too high nor too low

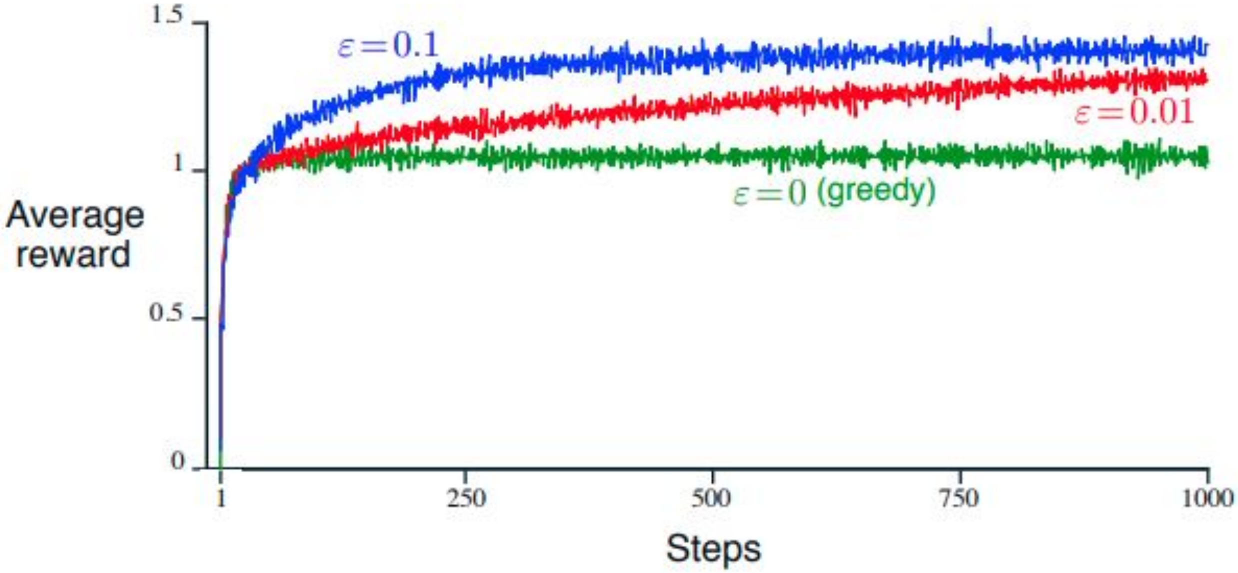
ε-greedy performance

Exploration parameters should usually be neither too high nor too low



ε-greedy performance

Exploration parameters should usually be neither too high nor too low



You will implement such an experiment in the assignments!

Action selection: exploration

We need to introduce exploration

Discuss three possible approaches:

1. Random perturbation (ε-greedy)
2. **Optimistic initialization** **(oi)**
3. Uncertainty-based (ucb)

Optimistic initialization

For learning based updates of the mean, the initial value really matters:

But: a higher initial value should also encourage exploration!

Optimistic initialization

For learning based updates of the mean, the initial value really matters:

But: a higher initial value should also encourage exploration!

Idea:

- Initialize optimistic mean estimates
- Select actions greedily
- Learning-based update of the mean

Optimistic initialization

Algorithm 3: Optimistic initialization with greedy action selection bandit algorithm.

Input: Initial value $\psi \in \mathbb{R}$, learning rate $\eta \in \mathbb{R}^+$, maximum number of timesteps T.

Initialization: Initialize $Q(a) = \psi \ \forall a \in \mathcal{A}$ `/* Optimistic init. */`

for $t = 1...T$ **do**

    $a_t = \arg\max_{a \in \mathcal{A}} Q(a)$ `/* Sample greedy action */`

    $r_t \sim p(r|a_t)$ `/* Sample reward */`

    $Q(a_t) \leftarrow Q(a_t) + \eta \cdot \left[r_t - Q(a_t)\right]$ `/* Learning update mean */`

end

Optimistic initialization

Algorithm 3: Optimistic initialization with greedy action selection bandit algorithm.

Input: $\boxed{\text{Initial value } \psi \in \mathbb{R}, \text{ learning rate } \eta \in \mathbb{R}^+}$, maximum number of timesteps T.

Initialization: Initialize $Q(a) = \psi \ \forall a \in \mathcal{A}$ `/* Optimistic init. */`

for $t = 1...T$ **do**

  $a_t = \arg\max_{a \in \mathcal{A}} Q(a)$ `/* Sample greedy action */`

  $r_t \sim p(r|a_t)$ `/* Sample reward */`

  $Q(a_t) \leftarrow Q(a_t) + \eta \cdot \left[r_t - Q(a_t)\right]$ `/* Learning update mean */`

end

Algorithm now requires an initial value ψ for each action (= exploration parameter) + a learning rate η

Optimistic initialization

Algorithm 3: Optimistic initialization with greedy action selection bandit algorithm.

Input: Initial value $\psi \in \mathbb{R}$, learning rate $\eta \in \mathbb{R}^+$, maximum number of timesteps T.

Initialization: $\boxed{\text{Initialize } Q(a) = \psi \ \forall a \in \mathcal{A}}$ `/* Optimistic init. */`

for $t = 1...T$ **do**

  $a_t = \arg\max_{a \in \mathcal{A}} Q(a)$ `/* Sample greedy action */`

  $r_t \sim p(r|a_t)$ `/* Sample reward */`

  $Q(a_t) \leftarrow Q(a_t) + \eta \cdot \Big[r_t - Q(a_t) \Big]$ `/* Learning update mean */`

end

Initialize all mean estimates to ψ

Optimistic initialization

Algorithm 3: Optimistic initialization with greedy action selection bandit algorithm.

Input: Initial value $\psi \in \mathbb{R}$, learning rate $\eta \in \mathbb{R}^+$, maximum number of timesteps T.

Initialization: Initialize $Q(a) = \psi \ \forall a \in \mathcal{A}$ `/* Optimistic init. */`

for $t = 1...T$ **do**

  $\boxed{a_t = \arg\max_{a \in \mathcal{A}} Q(a)}$ `/* Sample greedy action */`

  $r_t \sim p(r|a_t)$ `/* Sample reward */`

  $Q(a_t) \leftarrow Q(a_t) + \eta \cdot \Big[r_t - Q(a_t) \Big]$ `/* Learning update mean */`

end

Greedily select next action

Optimistic initialization

Algorithm 3: Optimistic initialization with greedy action selection bandit algorithm.

Input: Initial value $\psi \in \mathbb{R}$, learning rate $\eta \in \mathbb{R}^+$, maximum number of timesteps T.

Initialization: Initialize $Q(a) = \psi \; \forall a \in \mathcal{A}$ `/* Optimistic init. */`

for $t = 1 \ldots T$ **do**

$a_t = \arg\max_{a \in \mathcal{A}} Q(a)$ `/* Sample greedy action */`

$r_t \sim p(r|a_t)$ `/* Sample reward */`

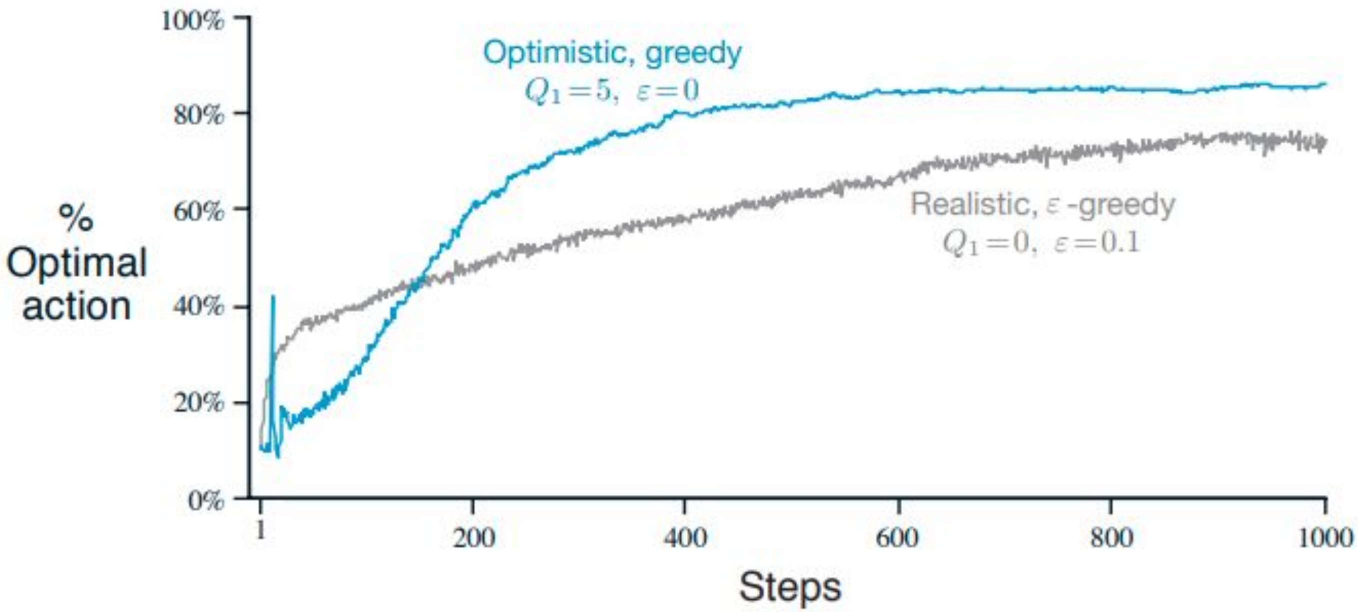
$\boxed{Q(a_t) \leftarrow Q(a_t) + \eta \cdot \Big[r_t - Q(a_t) \Big]}$ `/* Learning update mean */`

end

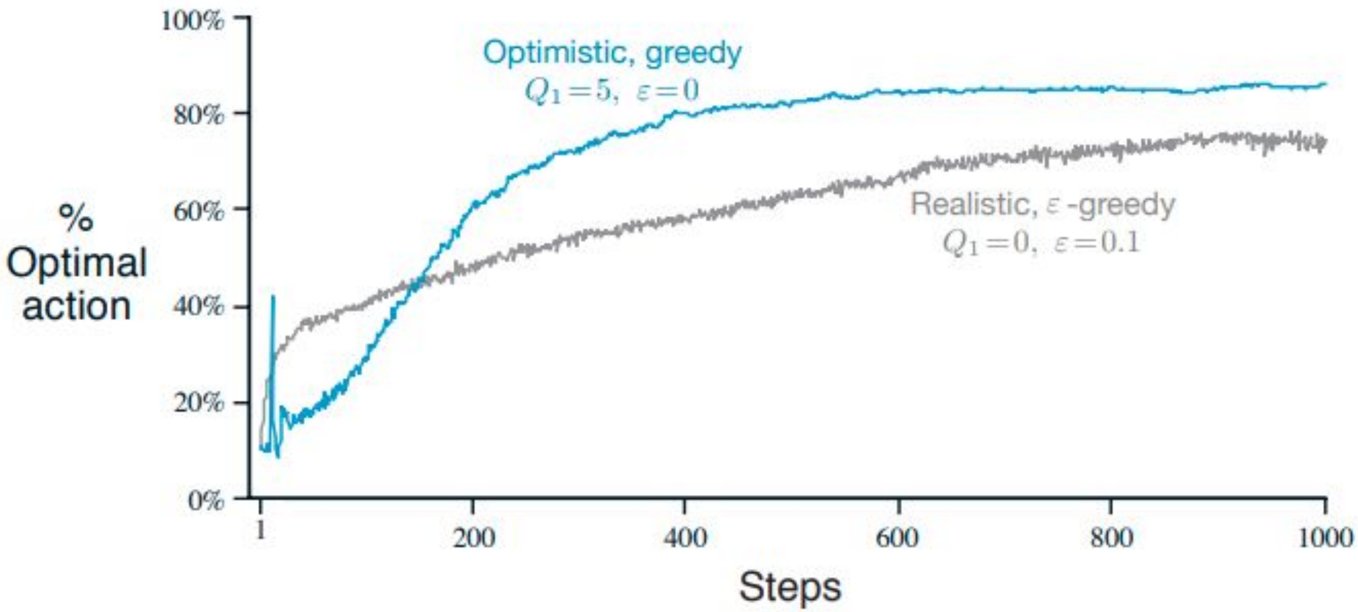
Learning update of means

Optimistic initialization performance

Optimistic initialization performance



Optimistic initialization performance



You will also experiment with different initial values in the assignment

Optimistic initialization performance

Question:

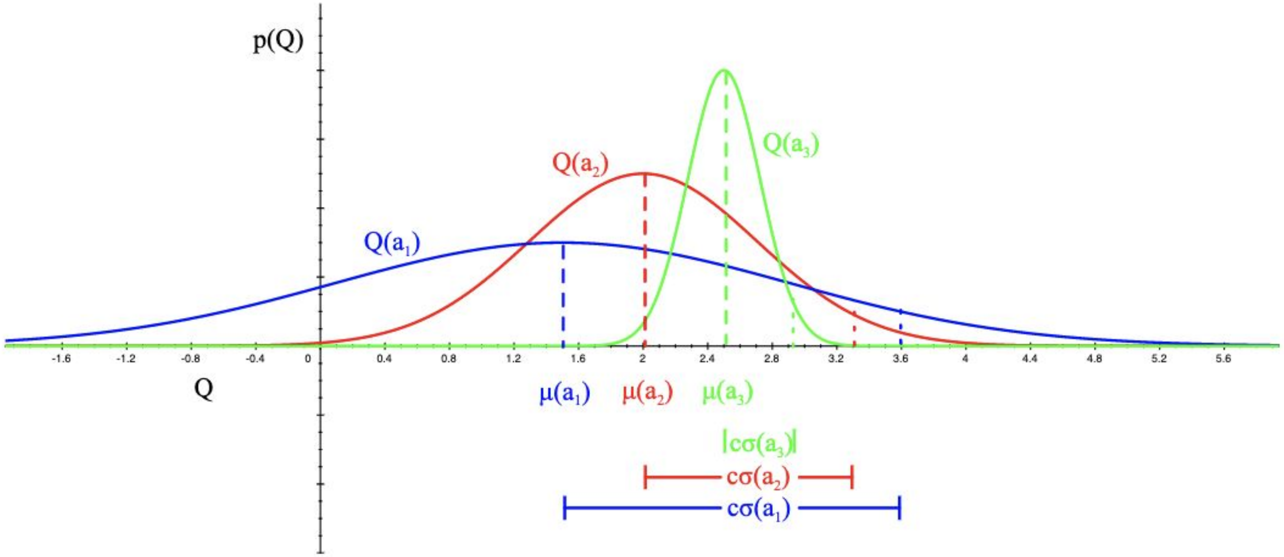
For the below bandit, what would be a good optimistic initial value?

A) 0

B) 4

C) 8

D) 1000



Optimistic initialization performance

Question:

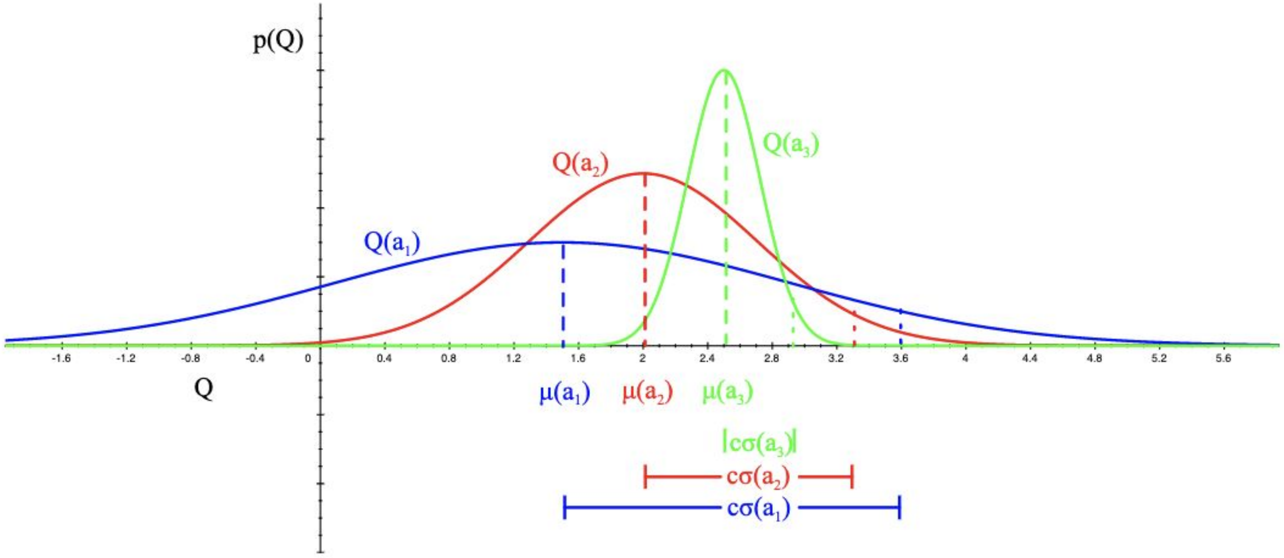
For the below bandit, what would be a good optimistic initial value?

A) 0

B) 4

C) 8

D) 1000



Optimistic initialization performance

Question:

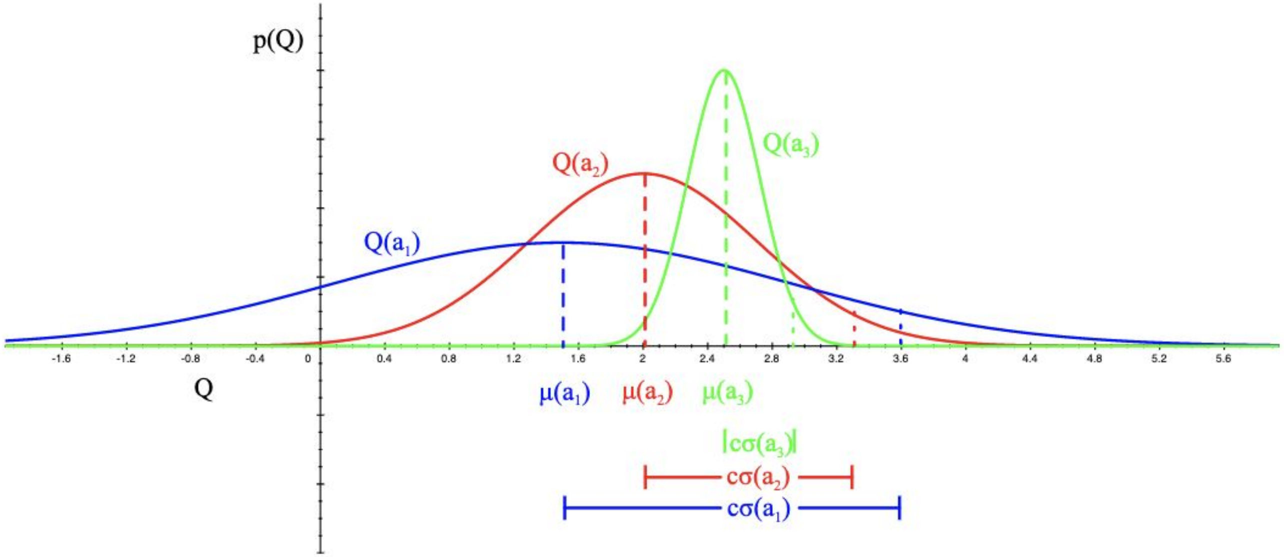
For the below bandit, what would be a good optimistic initial value?

A) 0

B) 4

C) 8

D) 1000



Put optimistic init neither too high nor too low

Action selection: exploration

We need to introduce exploration

Discuss three possible approaches:

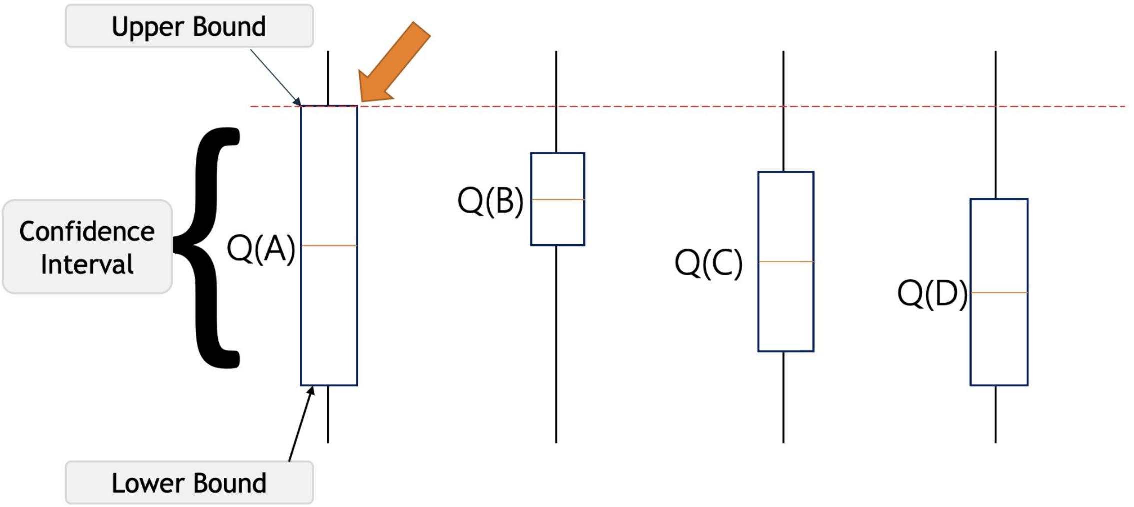
1. Random perturbation (ε-greedy)
2. Optimistic initialization (oi)
3. **Uncertainty-based (ucb)**

UCB

'If we could track the remaining uncertainty about each arm, we could more adaptively switch between exploration and exploitation'

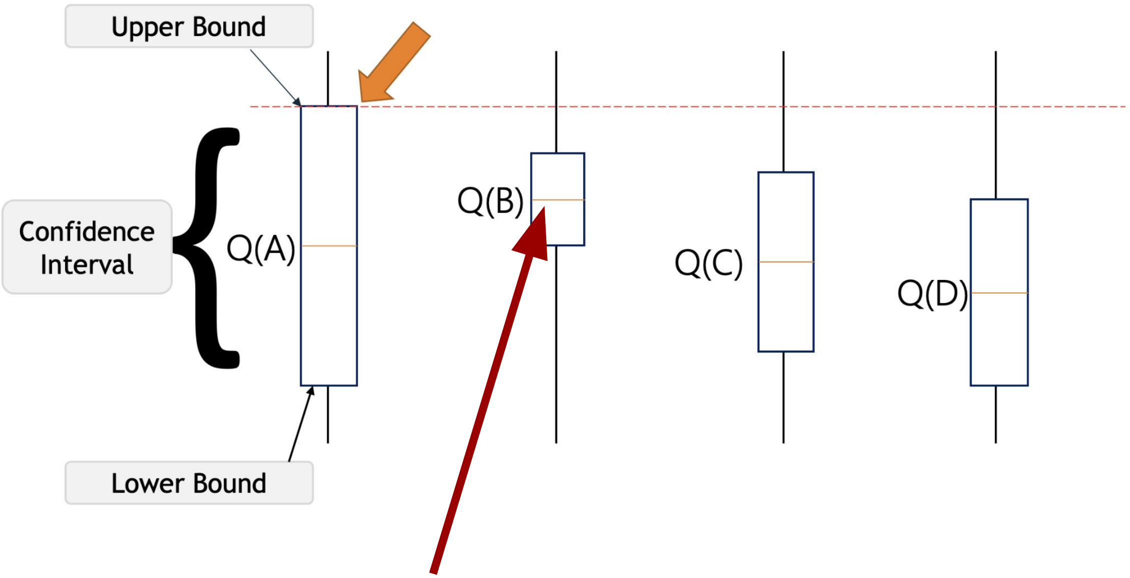
UCB

'If we could track the remaining uncertainty about each arm, we could more adaptively switch between exploration and exploitation'



UCB

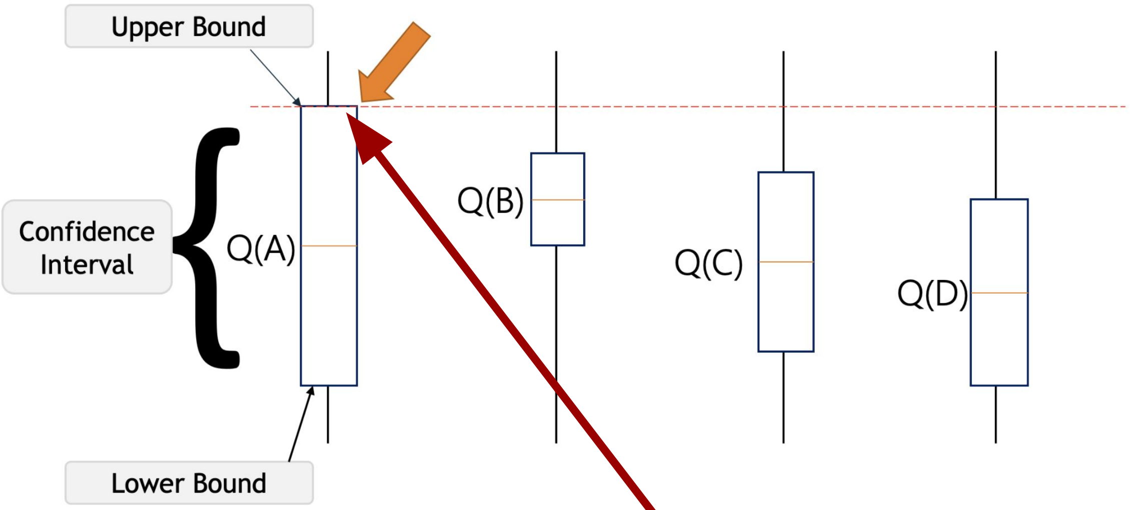
'If we could track the remaining uncertainty about each arm, we could more adaptively switch between exploration and exploitation'



Only considering the mean: action B seems most promising…

UCB

‘If we could track the remaining uncertainty about each arm, we could more adaptively switch between exploration and exploitation’



When we incorporate the uncertainty: action A may have more potential!

UCB

‘Optimism in the face of uncertainty’

UCB

'Optimism in the face of uncertainty'

- Estimate the mean and standard deviation of each action.
- Select action with *highest upper confidence bound* (UCB)

UCB

'Optimism in the face of uncertainty'

- Estimate the mean and standard deviation of each action.
- Select action with *highest upper confidence bound* (UCB)

$$a_{\mathrm{UCB}} = \arg\max_a \left[Q(a) + c \cdot \sqrt{\frac{\ln t}{n(a)}} \right]$$

UCB

'Optimism in the face of uncertainty'

- Estimate the mean and standard deviation of each action.
- Select action with *highest upper confidence bound* (UCB)

$$a_{\text{UCB}} = \arg\max_a \left[\boxed{Q(a)} + c \cdot \sqrt{\frac{\ln t}{n(a)}} \right]$$

Mean

UCB

'Optimism in the face of uncertainty'

- Estimate the mean and standard deviation of each action.
- Select action with *highest upper confidence bound* (UCB)

$$a_{\text{UCB}} = \arg\max_a \left[Q(a) + \boxed{c \cdot} \sqrt{\frac{\ln t}{n(a)}} \right]$$

c = exploration parameter

UCB

'Optimism in the face of uncertainty'

- Estimate the mean and standard deviation of each action.
- Select action with *highest upper confidence bound* (UCB)

$$a_{\text{UCB}} = \arg\max_a \left[Q(a) + c \cdot \boxed{\sqrt{\frac{\ln t}{n(a)}}} \right]$$

Approximately the standard error of mean
(decreases as square root of number of visits to action)

UCB

'Optimism in the face of uncertainty'

- Estimate the mean and standard deviation of each action.
- Select action with *highest upper confidence bound* (UCB)

$$a_{\text{UCB}} = \arg\max_a \left[Q(a) + c \cdot \sqrt{\frac{\ln t}{n(a)}} \right]$$

Important: when an action is untried (n(a)=0), we treat it's UCB estimate as ∞

UCB

'Optimism in the face of uncertainty'

- Estimate the mean and standard deviation of each action.
- Select action with *highest upper confidence bound* (UCB)

$$a_{\text{UCB}} = \arg\max_a \left[Q(a) + c \cdot \sqrt{\frac{\ln t}{n(a)}} \right]$$

Important: when an action is untried (n(a)=0), we treat it's UCB estimate as ∞

Effect: always prefer untried action over action that has been sampled

UCB pseudocode

Algorithm 4: UCB bandit algorithm.

Input: Exploration parameter $c \in \mathbb{R}^+$, maximum number of timesteps T.

Initialization: Initialize $Q(a) = 0$, $n(a) = 0 \ \forall a \in \mathcal{A}$

for $t = 1...T$ **do**

$a_t = \arg\max_a \left[Q(a) + c \cdot \sqrt{\frac{\ln t}{n(a)}} \right]$ `/* UCB action */`

$r_t \sim p(r|a_t)$ `/* Sample reward */`

$n(a_t) \leftarrow n(a_t) + 1$ `/* Update count */`

$Q(a_t) \leftarrow Q(a_t) + \frac{1}{n(a_t)} \left[r_t - Q(a_t) \right]$ `/* Incr. update mean */`

end

UCB pseudocode

Algorithm 4: UCB bandit algorithm.

Input: $\boxed{\text{Exploration parameter } c \in \mathbb{R}^+}$, maximum number of timesteps T.

Initialization: Initialize $Q(a) = 0$, $n(a) = 0 \ \forall a \in \mathcal{A}$

for $t = 1...T$ **do**

$a_t = \arg\max_a \left[Q(a) + c \cdot \sqrt{\frac{\ln t}{n(a)}} \right]$ `/* UCB action */`

$r_t \sim p(r|a_t)$ `/* Sample reward */`

$n(a_t) \leftarrow n(a_t) + 1$ `/* Update count */`

$Q(a_t) \leftarrow Q(a_t) + \frac{1}{n(a_t)} \left[r_t - Q(a_t) \right]$ `/* Incr. update mean */`

end

Needs exploration parameter c as input (higher c = more exploration)

UCB pseudocode

Algorithm 4: UCB bandit algorithm.

Input: Exploration parameter $c \in \mathbb{R}^+$, maximum number of timesteps T.

Initialization: $\boxed{\text{Initialize } Q(a) = 0,\ n(a) = 0\ \forall a \in \mathcal{A}}$

for $t = 1...T$ **do**

$a_t = \arg\max_a \left[Q(a) + c \cdot \sqrt{\frac{\ln t}{n(a)}} \right]$ `/* UCB action */`

$r_t \sim p(r|a_t)$ `/* Sample reward */`

$n(a_t) \leftarrow n(a_t) + 1$ `/* Update count */`

$Q(a_t) \leftarrow Q(a_t) + \frac{1}{n(a_t)} \left[r_t - Q(a_t) \right]$ `/* Incr. update mean */`

end

Initialize means and counts to 0

UCB pseudocode

Algorithm 4: UCB bandit algorithm.

Input: Exploration parameter $c \in \mathbb{R}^+$, maximum number of timesteps T.

Initialization: Initialize $Q(a) = 0$, $n(a) = 0 \; \forall a \in \mathcal{A}$

for $t = 1...T$ **do**

$\boxed{a_t = \arg\max_a \left[Q(a) + c \cdot \sqrt{\frac{\ln t}{n(a)}} \right]}$ `/* UCB action */`

$r_t \sim p(r|a_t)$ `/* Sample reward */`

$n(a_t) \leftarrow n(a_t) + 1$ `/* Update count */`

$Q(a_t) \leftarrow Q(a_t) + \frac{1}{n(a_t)} \left[r_t - Q(a_t) \right]$ `/* Incr. update mean */`

end

Use UCB formula for action selection

UCB pseudocode

Algorithm 4: UCB bandit algorithm.

Input: Exploration parameter $c \in \mathbb{R}^+$, maximum number of timesteps T.

Initialization: Initialize $Q(a) = 0$, $n(a) = 0 \; \forall a \in \mathcal{A}$

for $t = 1 \ldots T$ **do**

$a_t = \arg\max_a \left[Q(a) + c \cdot \sqrt{\frac{\ln t}{n(a)}} \right]$ `/* UCB action */`

$r_t \sim p(r|a_t)$ `/* Sample reward */`

$n(a_t) \leftarrow n(a_t) + 1$ `/* Update count */`

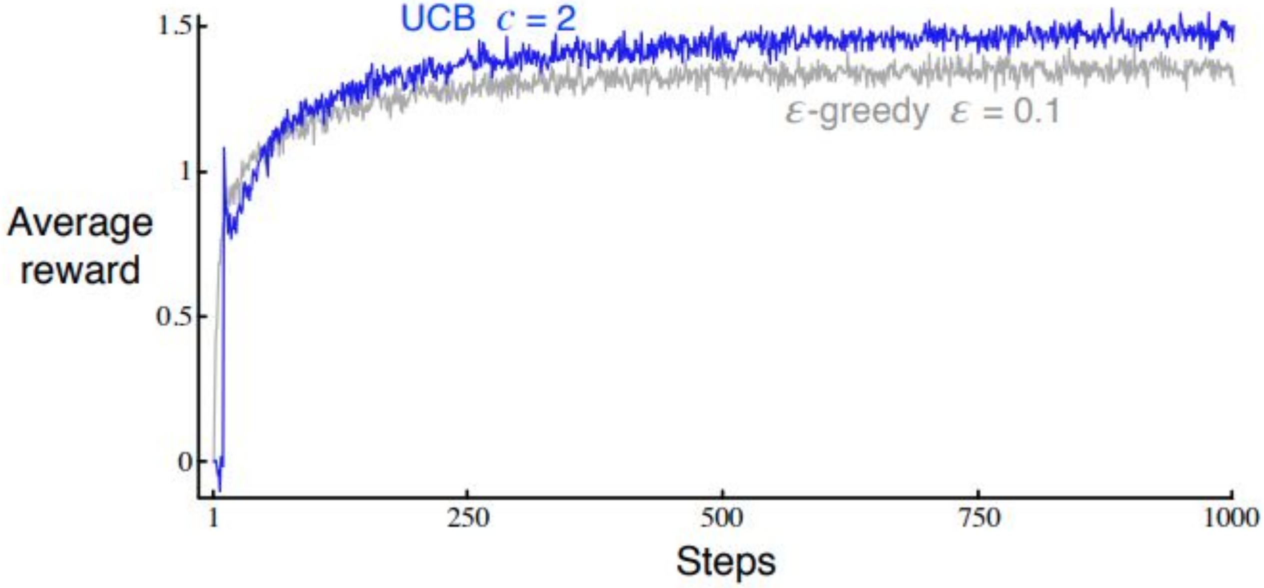
$\boxed{Q(a_t) \leftarrow Q(a_t) + \frac{1}{n(a_t)} \left[r_t - Q(a_t) \right]}$ `/* Incr. update mean */`

end

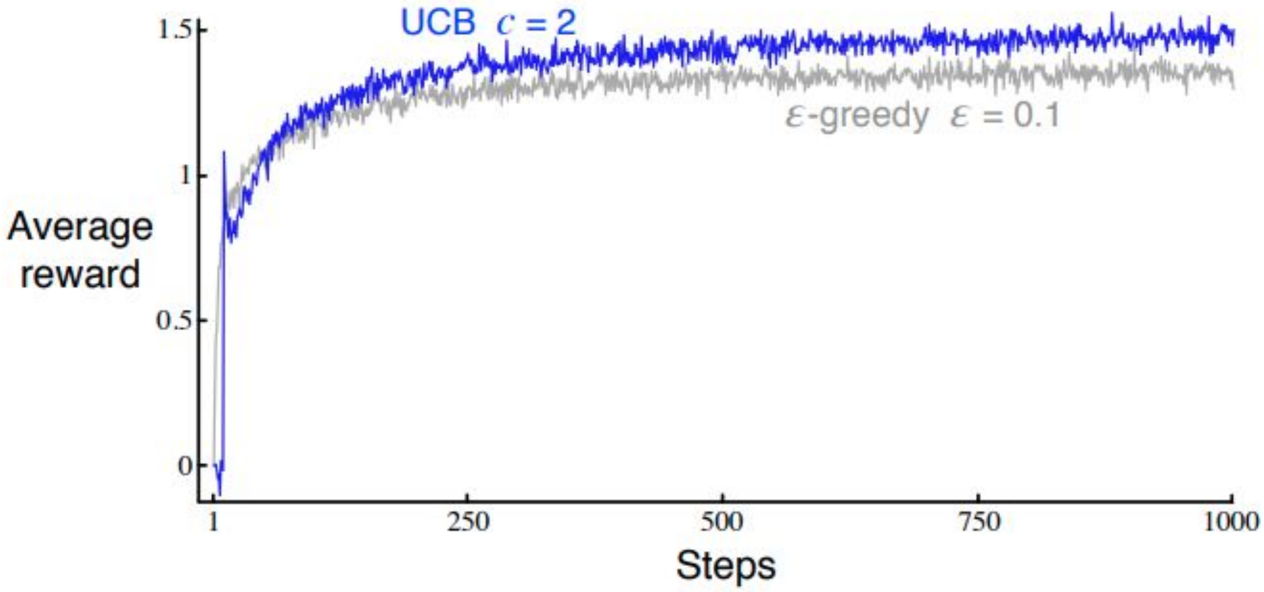
Incremental update of the mean

UCB performance

UCB performance



UCB performance



Typically does better than the other two:

more gradual switch from exploration to exploitation

Advanced bandit topics

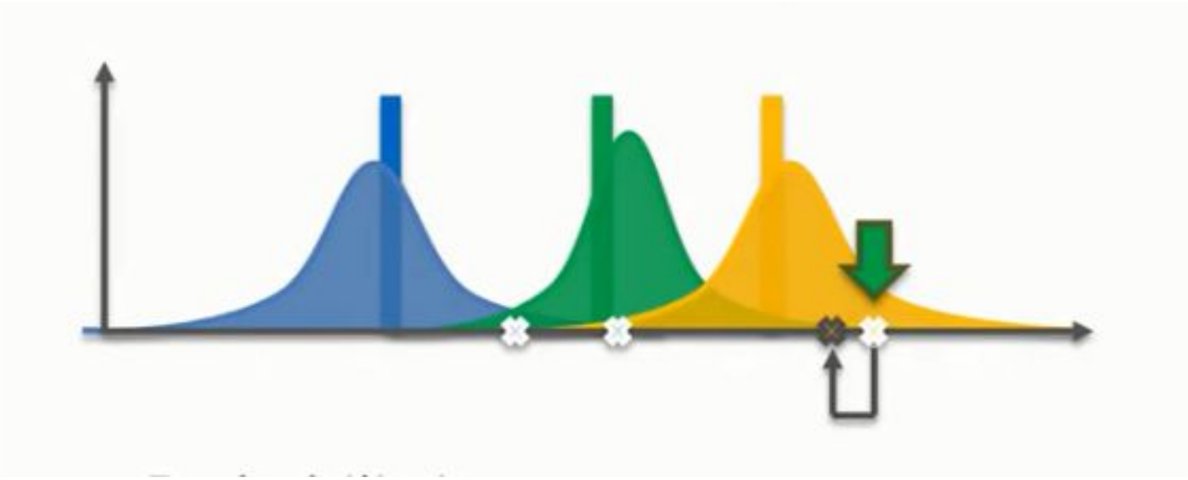
- Much research, many more exploration strategies.
    - e.g. Bayesian methods:

Advanced bandit topics

- Much research, many more exploration strategies.
    - e.g. Bayesian methods:
        1. Estimate a posterior distribution for each mean estimate

Advanced bandit topics

- Much research, many more exploration strategies.
  - e.g. Bayesian methods:
    1. Estimate a posterior distribution for each mean estimate
    2. Use Thompson sampling (probability matching):
       select action according to probability that it is the best one



Advanced bandit topics

- Much research, many more exploration strategies.
  - e.g. Bayesian methods:
    1. Estimate a posterior distribution for each mean estimate
    2. Use Thompson sampling (probability matching):
       select action according to probability that it is the best one

Very theoretical branch of research (more than RL):

proving convergence properties of algorithms.

Part 5:

Contextual bandits & MDPs

Contextual bandit

Often the reward distribution of the bandit you face depends on *context*

Contextual bandit

Often the reward distribution of the bandit you face depends on *context*

Contextual bandit

Often the reward distribution of the bandit you face depends on *context*

Contextual bandit: reward distribution depends on <u>context state s</u>
(like age and gender of user in advertisement recommendations)

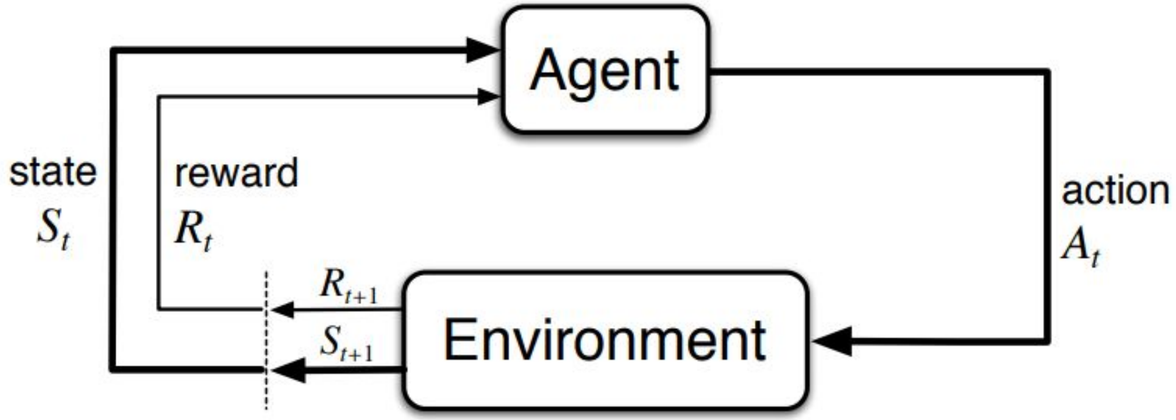
Markov Decision Process

When the state also changes based on our action we call it a

Markov Decision Process (MDP)

Markov Decision Process

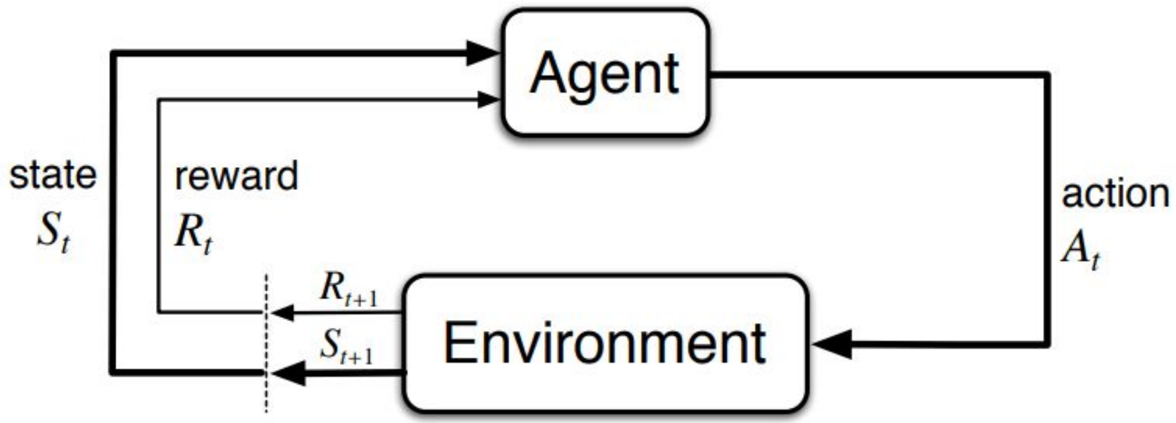
When the state also changes based on our action we call it a
Markov Decision Process (MDP)



Markov Decision Process

When the state also changes based on our action we call it a

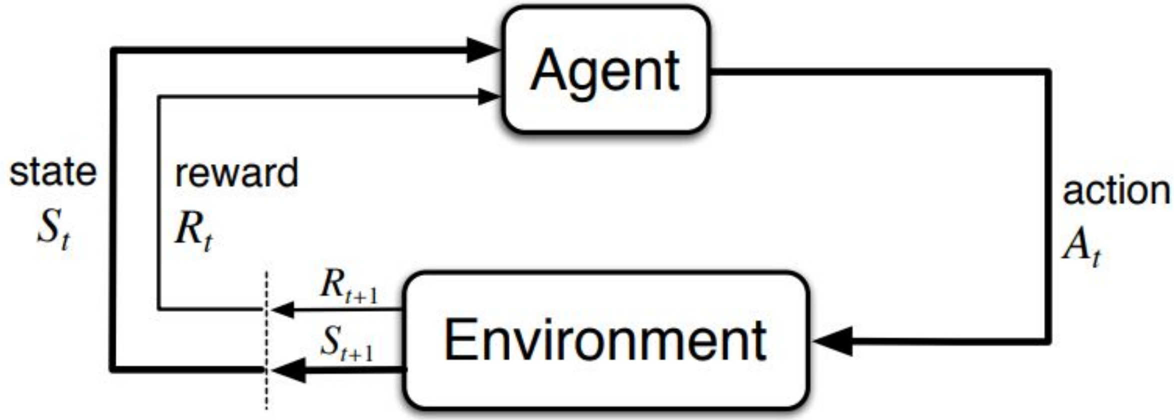
Markov Decision Process (MDP)



Framework underneath reinforcement learning
(where exploration/exploitation is just as important)

Markov Decision Process

When the state also changes based on our action we call it a
Markov Decision Process (MDP)



Next week!

Framework underneath reinforcement learning
(where exploration/exploitation is just as important)

To Do

Read:

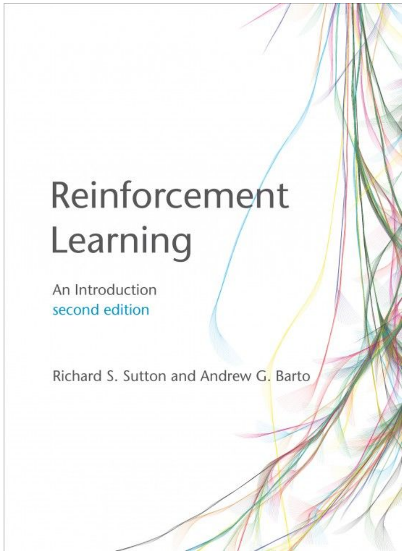
Assignment:

To Do

Read:

1. Sutton & Barto, Chapter 2 (multi-armed bandit)
2. Lecture slides and lecture notes

Assignment:



Free online version:
http://incompleteideas.net/book/RLbook2020.pdf

To Do

Read:

1. Sutton & Barto, Chapter 2 (multi-armed bandit)
2. Lecture slides and lecture notes

Assignment:

1. Implement three bandit algorithms:
    - ε-greedy
    - Optimistic initialization
    - UCB

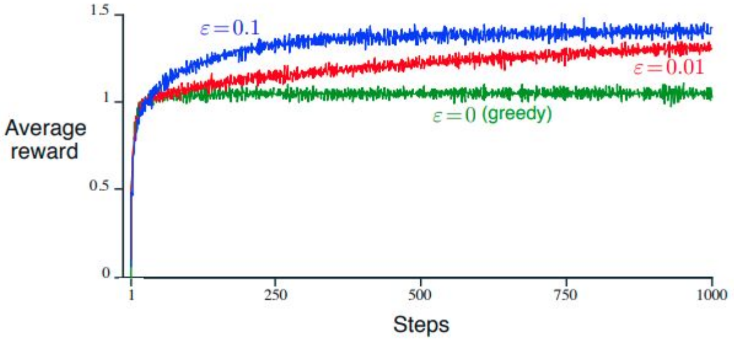
To Do

Read:

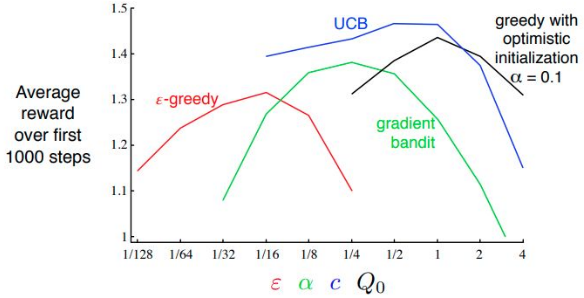
1. Sutton & Barto, Chapter 2 (multi-armed bandit)
2. Lecture slides and lecture notes

Assignment:

1. Implement three bandit algorithms:
   - ε-greedy
   - Optimistic initialization
   - UCB



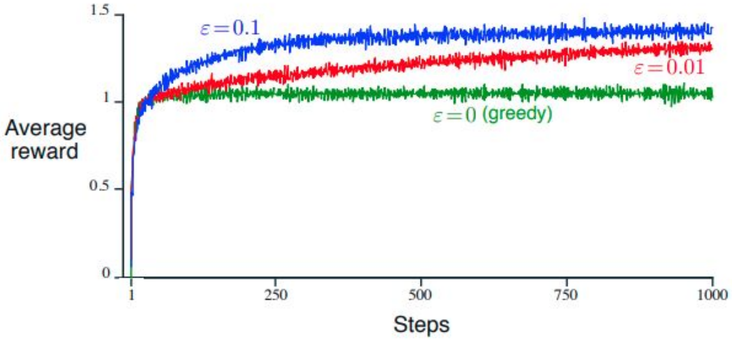
2. Compare their performance



To Do

Read:

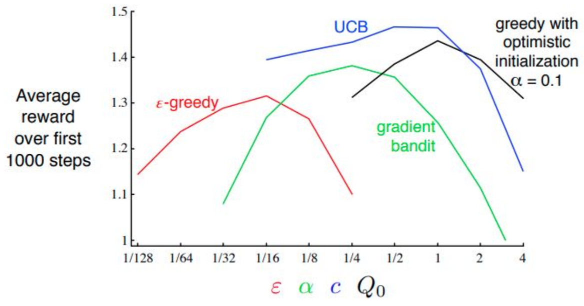
1. Sutton & Barto, Chapter 2 (multi-armed bandit)
2. Lecture slides and lecture notes



Assignment:

1. Implement three bandit algorithms:
   - ε-greedy
   - Optimistic initialization
   - UCB

2. Compare their performance



3. Write a report

Questions?