Deep Reinforcement Learning

2022-23 Second Semester, M.Tech (AIML)

Session #2-3:
Multi-armed Bandits

Instructors :

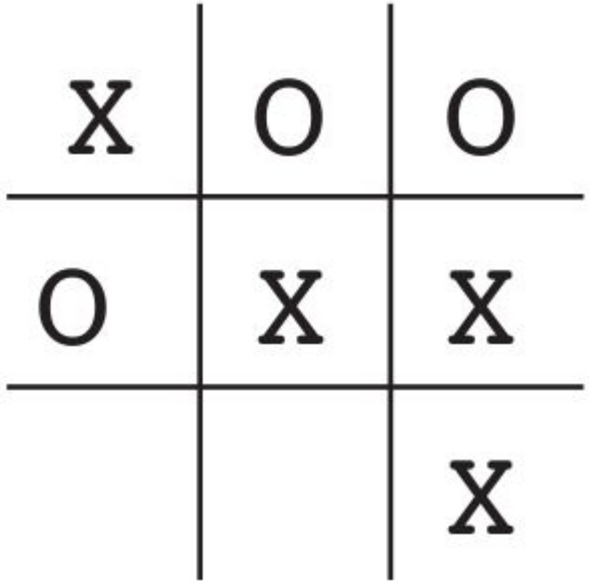
1. Prof. S. P. Vimal (vimalsp@wilp.bits-pilani.ac.in),
2. Dr. V Chandra Sekhar (chandrasekhar.v@wilp.bits-pilani.ac.in)

Agenda for the class

- Recap
- k-armed Bandit Problem & its significance
- Action-Value Methods
  - Sample Average Method & Incremental Implementation
- Non-stationary Problem
- Initial Values & Action Selection
- Gradient Bandit Algorithms [Class #3]
- Associative Search [Class #3]

Acknowledgements: Some of the slides were adopted *with permission* from the course CSCE-689 (Texas A&M University) by Prof. Guni Sharon

Tic-Tac-Toc



Tic-Tac-Toc

States	Initial Values
$\begin{bmatrix} X & & \\ & & \\ & & \end{bmatrix}$	0.5
$\begin{bmatrix} X & O & O \\ & X & \\ & & \end{bmatrix}$	0.5
$\begin{bmatrix} X & O & O \\ & X & \\ & & X \end{bmatrix}$	1.0
$\begin{bmatrix} X & & O \\ X & & O \\ & X & O \end{bmatrix}$	0
...	...

Learning Task: Play as many times against the opponent and learn the values

X	O	O
O	X	X
		X

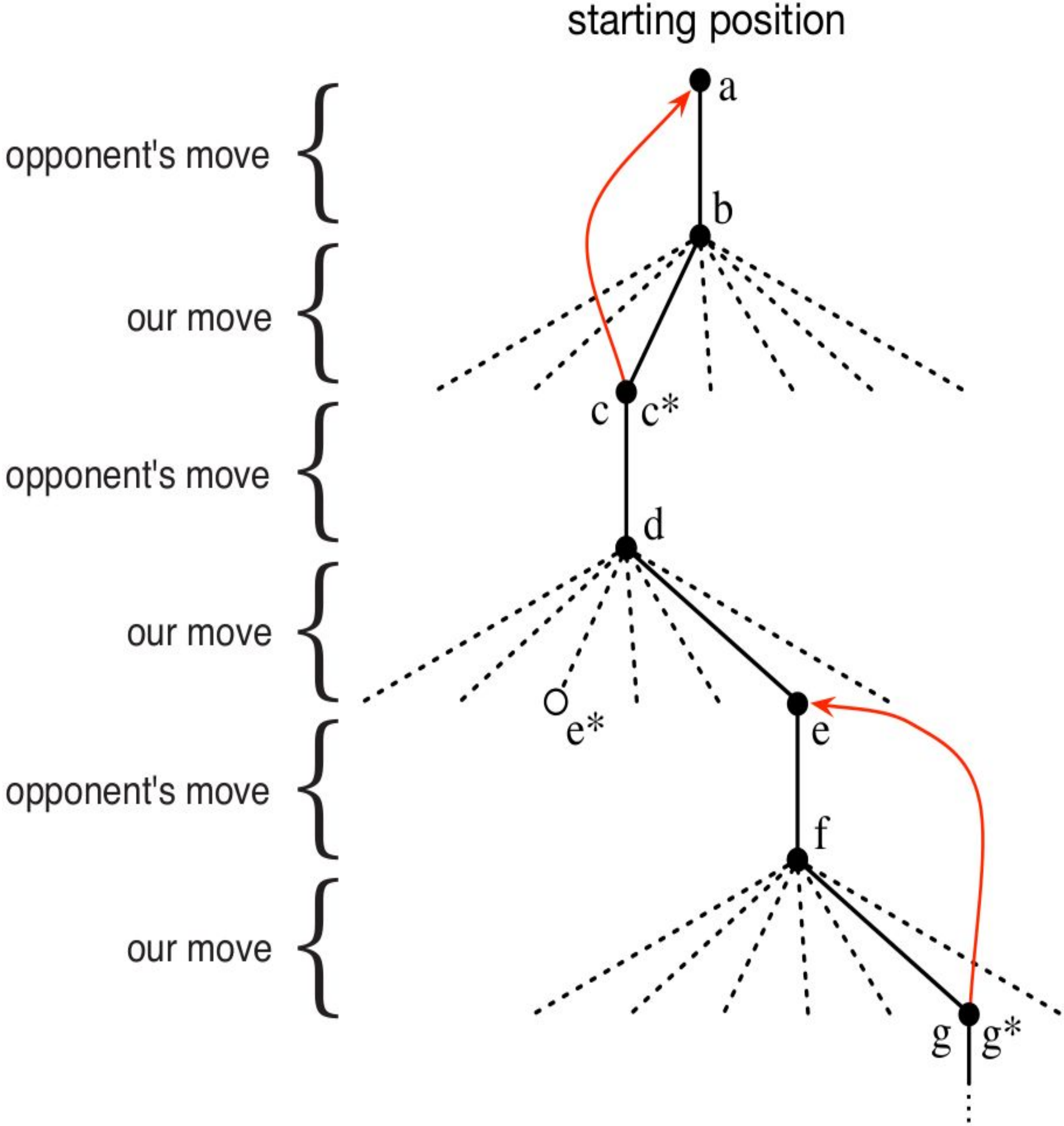
Set up a table of states initial values

Tic-Tac-Toc (prev. class)

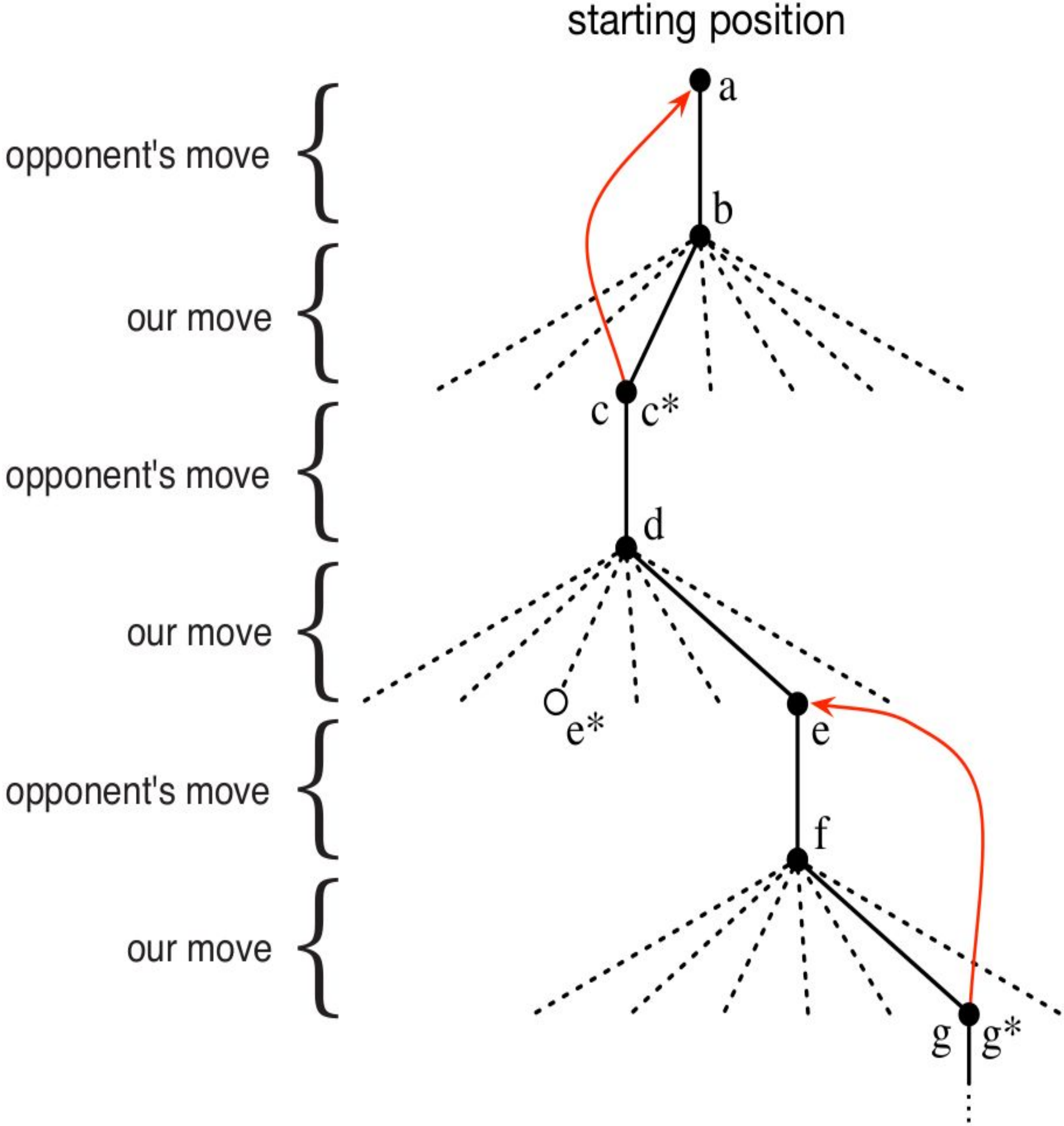
States	Initial Values
$\begin{bmatrix} X & & \\ & & \\ & & \end{bmatrix}$	**0.5**
$\begin{bmatrix} X & O & O \\ & X & \\ & & \end{bmatrix}$	**0.5**
$\begin{bmatrix} \boldsymbol{X} & O & O \\ & \boldsymbol{X} & \\ & & \boldsymbol{X} \end{bmatrix}$	**1.0**
$\begin{bmatrix} X & & O \\ \boldsymbol{X} & & O \\ & \boldsymbol{X} & O \end{bmatrix}$	**0**
...	**...**

S_t - state before greedy move
S_{t+1} - state after greedy move

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[V(S_{t+1}) - V(S_t)\Big]$$



Tic-Tac-Toc (prev. class)



Temporal Difference Learning Rule

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[V(S_{t+1}) - V(S_t)\Big]$$

***α*- Step Size Parameter**

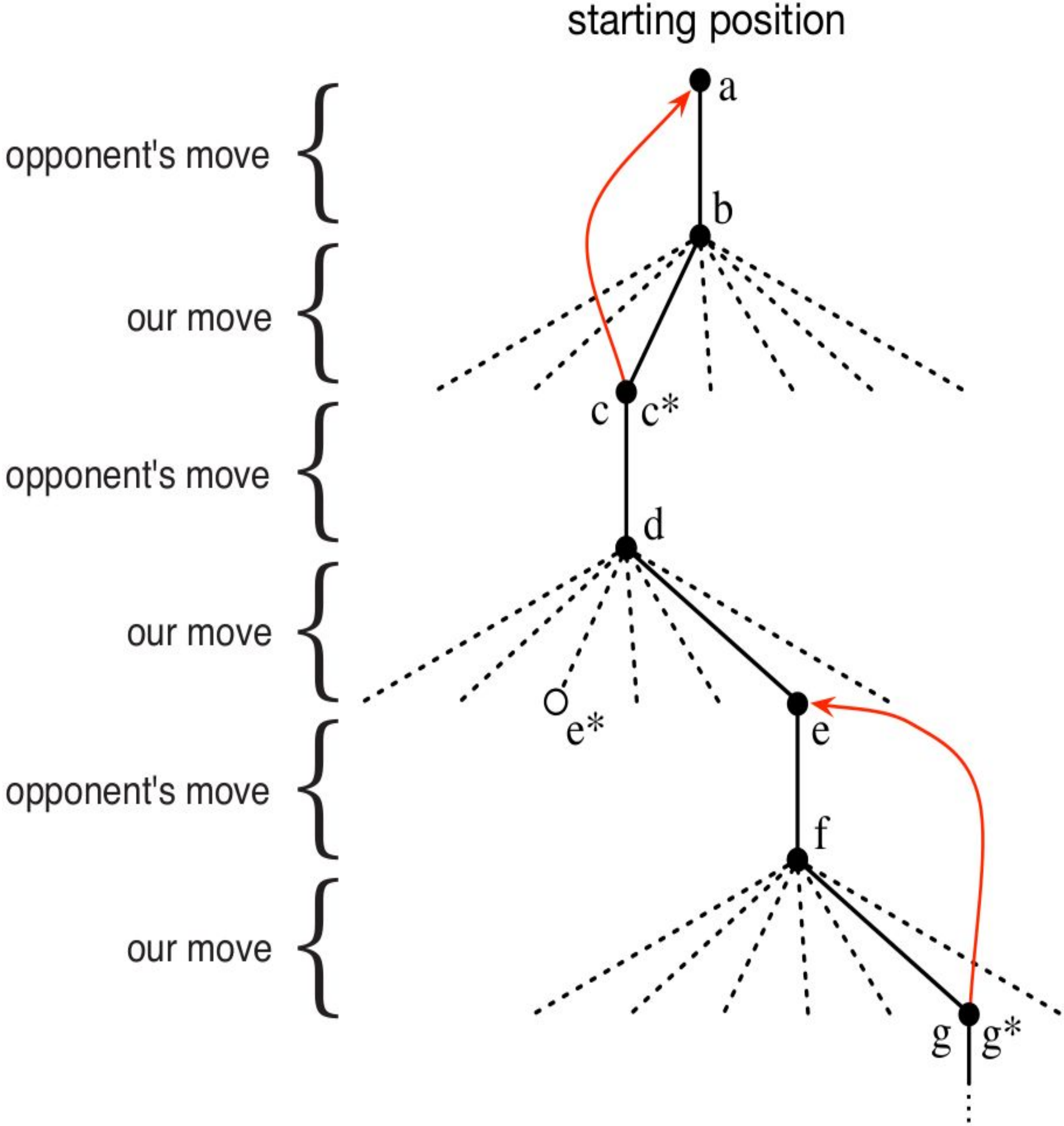
Tic-Tac-Toc (prev. class)

Questions:

(1) What happens if α is gradually made to 0 over many games with the opponent?

(2) What happens if α is gradually reduced over many games, but never made 0?

(3) What happens if α is kept constant throughout its life time?



Temporal Difference Learning Rule

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[V(S_{t+1}) - V(S_t)\Big]$$

α- Step Size Parameter

Tic-Tac-Toc (prev. class)

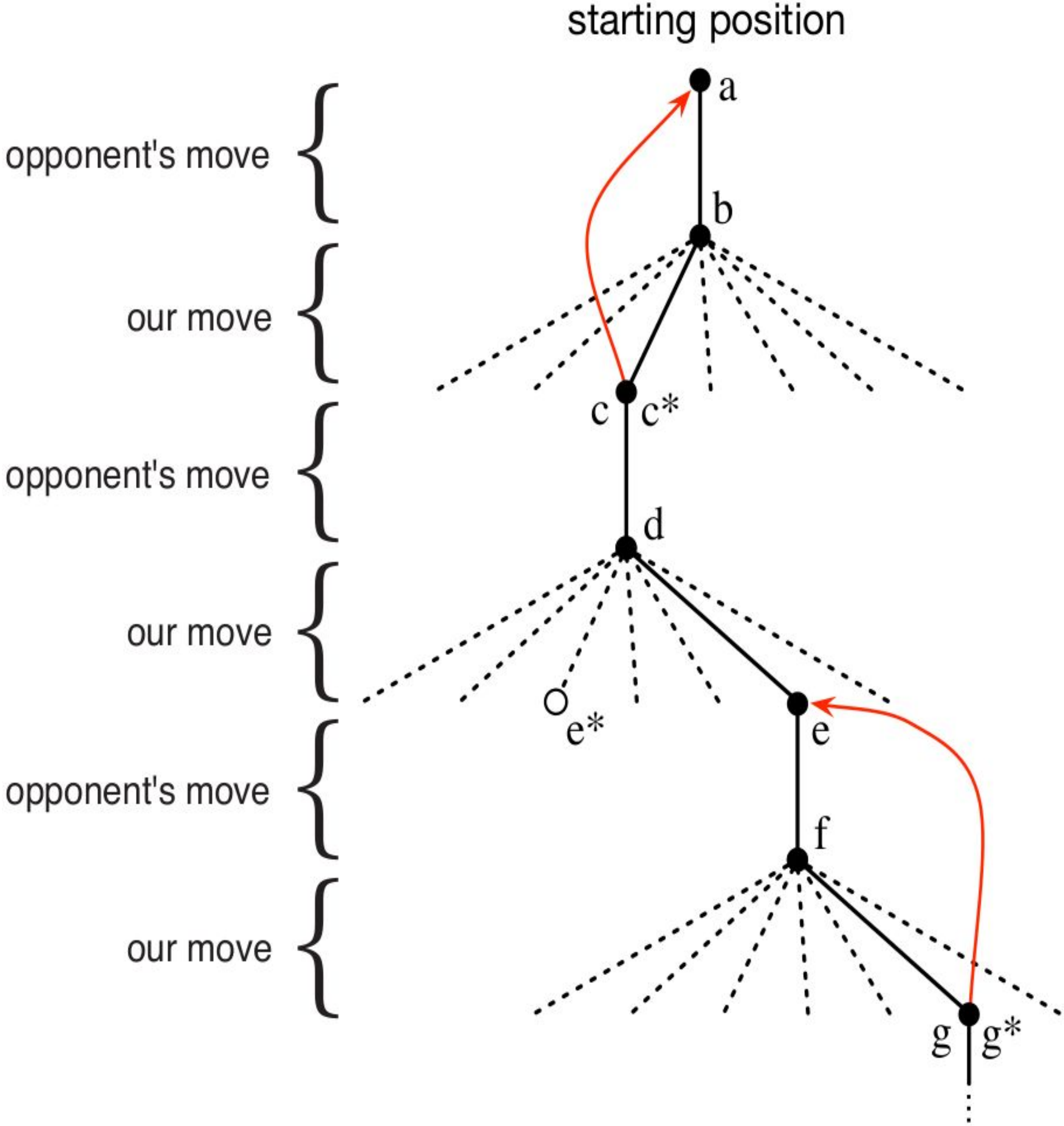
Key Takeaways:

(1) Learning while interacting with the environment (opponent).

(2) We have a clear goal

(3) Our policy is to make moves that maximizes our chances of reaching goal

- Use the values of states most of the time (exploration) and explore rest of the time.



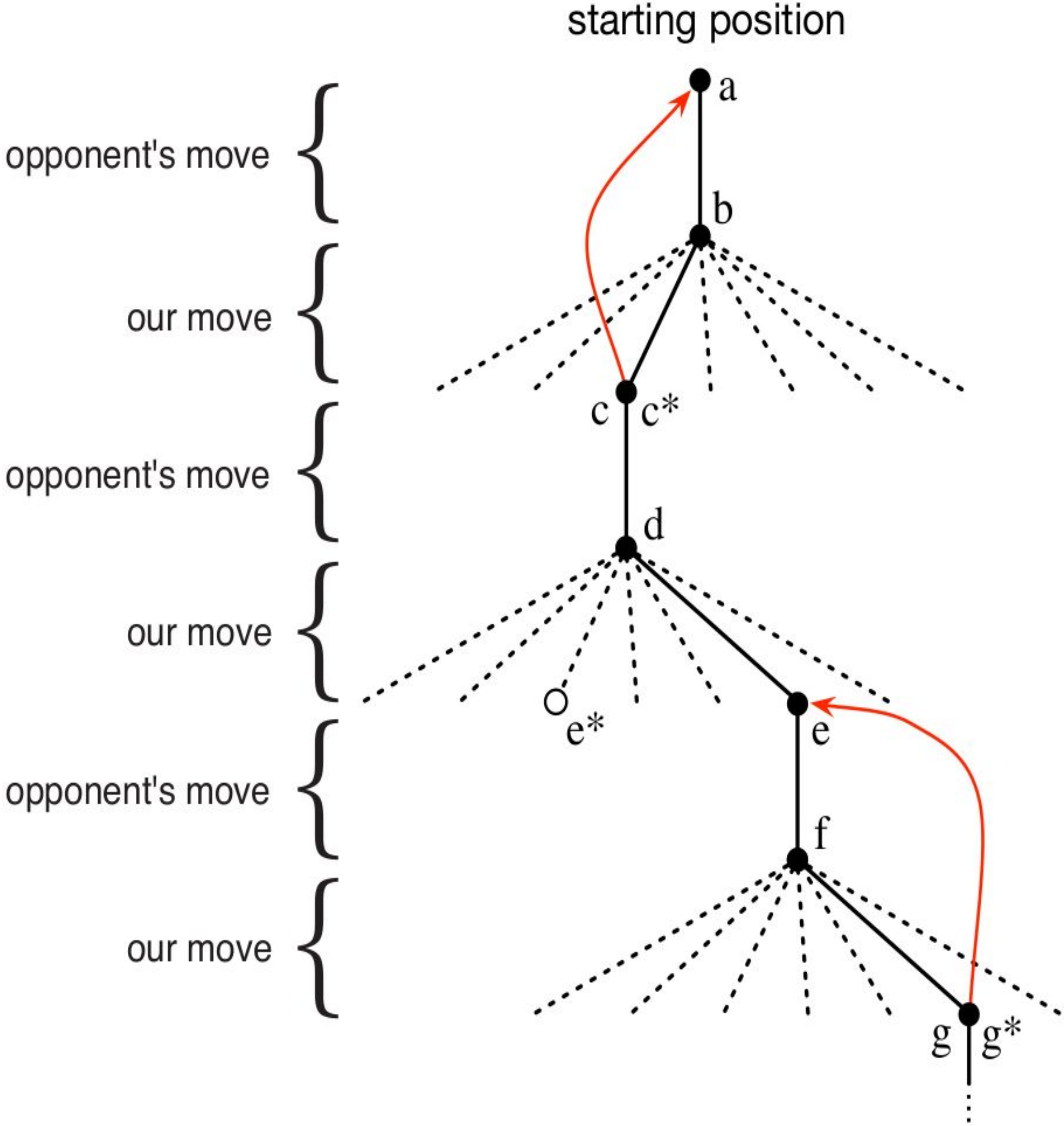
Temporal Difference Learning Rule

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[V(S_{t+1}) - V(S_t)\Big]$$

α- Step Size Parameter

Tic-Tac-Toc (prev. class)

Reading Assigned:
Identify how this reinforcement learning solution is different from solutions using minimax algorithm and genetic algorithms.
Post your answers in the discussion forum;



Temporal Difference Learning Rule

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[V(S_{t+1}) - V(S_t)\Big]$$

α- Step Size Parameter

K-armed Bandit Problem

K-armed Bandit Problem

Problem

- You are faced repeatedly with a choice among k different options, or actions
- After each choice of actions you receive a numerical reward

  Reward is chosen from a stationary probability distribution that depends on the selected action
- **Objective** : to *maximize the expected total reward over some time period*

K-armed Bandit Problem

Problem

- You are faced repeatedly with a choice among k different options, or actions
- After each choice of actions you receive a numerical reward

  Reward is chosen from a stationary probability distribution that depends on the selected action
- **Objective** : to *maximize the expected total reward over some time period*




K-armed Bandit Problem

Problem

- You are faced repeatedly with a choice among k different options, or actions
- After each choice of actions you receive a numerical reward

    Reward is chosen from a stationary probability distribution that depends on the selected action

- **Objective** : to *maximize the expected total reward over some time period*

<u>Strategy</u>:

- Identify the best lever(s)
- Keep pulling the identified ones

<u>Questions</u>:

- How do we define the *best ones*?
- What are the best levers?

K-armed Bandit Problem

Problem

- You are faced repeatedly with a choice among k different options, or actions
- After each choice of actions you receive a numerical reward

  Reward is chosen from a stationary probability distribution that depends on the selected action

- **Objective** : to *maximize the expected total reward over some time period*



Strategy:

- Identify the best lever(s)
- Keep pulling the identified ones

Questions:

- How do we define the *best ones*?
- What are the best levers?

K-armed Bandit Problem

Problem

- You are faced repeatedly with a choice among k different options, or actions
- After each choice of actions you receive a numerical reward

  Reward is chosen from a stationary probability distribution that depends on the selected action
- **Objective** : to *maximize the expected total reward over some time period*

$\mathbb{E}[a] = -\$0.5$

$\mathbb{E}[b] = -\$0.2$

$\mathbb{E}[c] = \$0.1$

$\mathbb{E}[d] = \$0.11$

- ***Expected Mean Reward*** for each action selected
  → call it ***Value*** of the action

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

K-armed Bandit Problem

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

- A_t - action selected on time step t
- Q_t (a) - estimated value of action a at time step t
- q_*(a) - value of an arbitrary action a

Note: If you knew the value of each action, then it would be trivial to solve the k -armed bandit problem: you would always select the action with highest value :-)

K-armed Bandit Problem

$\mathbb{E}[a] = -\$0.5$

$\mathbb{E}[b] = -\$0.2$

$\mathbb{E}[c] = \$0.1$

$\mathbb{E}[d] = \$0.11$

K-armed Bandit Problem

-1, -1, 5

$\hat{\mathbb{E}}[a]=1$

-0.2, -0.2

$\hat{\mathbb{E}}[b]=-0.2$

-0.5, -0.5, -0.5

$\hat{\mathbb{E}}[c]=-0.5$

-2, -2

$\hat{\mathbb{E}}[d]=-2$

Keep pulling the levers; update the estimate of action values;

K-armed Bandit Problem

-1, -1, 5

$\hat{\mathbb{E}}[a]=1$

-0.2, -0.2

$\hat{\mathbb{E}}[b]=-0.2$

-0.5, -0.5, -0.5

$\hat{\mathbb{E}}[c]=-0.5$

-2, -2

$\hat{\mathbb{E}}[d]=-2$

K-armed Bandit Problem

1. How to maintain the estimate of expected rewards for each action?

   Average the rewards actually received !!!

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$$

1. How to use the estimate in selecting the right action?

   <u>Greedy Action Selection</u> $$A_t \doteq \underset{a}{\arg\max}\, Q_t(a)$$

K-armed Bandit Problem

2. How to use the estimate in selecting the right action?

Greedy Action Selection

$$A_t \doteq \underset{a}{\arg\max}\, Q_t(a)$$

Actions which are inferior by the value estimate upto time t, could be indeed better than the greedy action at t !!!

3. Exploration vs. Exploitation?

ε-Greedy Action Selection / near-greedy action selection

Behave greedily most of the time; Once in a while, with small probability ε select randomly from among all the actions with equal probability, independently of the action-value estimates.

K-armed Bandit Problem

{-1, -1, 5}	{-0.2, -0.2}	{-0.5, -0.5, -0.5}	{-2, -2}
$N_{11}(a)=3$	$N_{11}(b)=2$	$N_{11}(c)=3$	$N_{11}(d)=2$
$q_*(a)=-\$0.5$	$q_*(b)=-\$0.2$	$q_*(c)=\$0.1$	$q_*(d)=\$1$
$Q_{11}(a)=1$	$Q_{11}(b)=-0.2$	$Q_{11}(c)=-0.5$	$Q_{11}(d)=-2$

K-armed Bandit Problem

Greedy Action

{-1, -1, 5}

$N_{11}(a)=3$

$q_*(a)=-\$0.5$

$Q_{11}(a)=1$

{-0.2, -0.2}

$N_{11}(b)=2$

$q_*(b)=-\$0.2$

$Q_{11}(b)=-0.2$

{-0.5, -0.5, -0.5}

$N_{11}(c)=3$

$q_*(c)=\$0.1$

$Q_{11}(c)=-0.5$

{-2, -2}

$N_{11}(d)=2$

$q_*(d)=\$1$

$Q_{11}(d)=-2$

K-armed Bandit Problem

Action to Explore

{-1, -1, 5}

$N_{11}(a)=3$

$q_*(a)=-\$0.5$

$Q_{11}(a)=1$

{-0.2, -0.2}

$N_{11}(b)=2$

$q_*(b)=-\$0.2$

$Q_{11}(b)=-0.2$

{-0.5, -0.5, -0.5}

$N_{11}(c)=3$

$q_*(c)=\$0.1$

$Q_{11}(c)=-0.5$

{-2, -2}

$N_{11}(d)=2$

$q_*(d)=\$1$

$Q_{11}(d)=-2$

K-armed Bandit Problem

ε-Greedy Action Selection / near-greedy action selection

```
epsilon = 0.05 // small value to control exploration
def get_action():
    if random.random() > epsilon:
        return argmaxa(Q(a))
    else:
        return random.choice(A)
```

- **In the limit as the number of steps increases, every action will be sampled by ε-greedy action selection an infinite number of times. This ensures that all the Qt (a) converge to q_* (a).**
- **Easy to implement / optimize for epsilon / yields good results**

Ex-1: In ε-greedy action selection, for the case of two actions and ε = 0.5, what is the probability that *the greedy action* is selected?

Ex-1: In ε-greedy action selection, for the case of two actions and ε = 0.5, what is the probability that *the greedy action* is selected?

p (greedy action)

= p (greedy action AND greedy selection) + p (greedy action AND random selection)

= p (greedy action | greedy selection) p (greedy selection)

+ p (greedy action | random selection) p (random selection)

= p (greedy action | greedy selection) (1-**ε**) + p (greedy action | random selection) (**ε**)

= p (greedy action | greedy selection) (0.5) + p (greedy action | random selection) (0.5)

= (1) (0.5) + (0.5) (0.5)

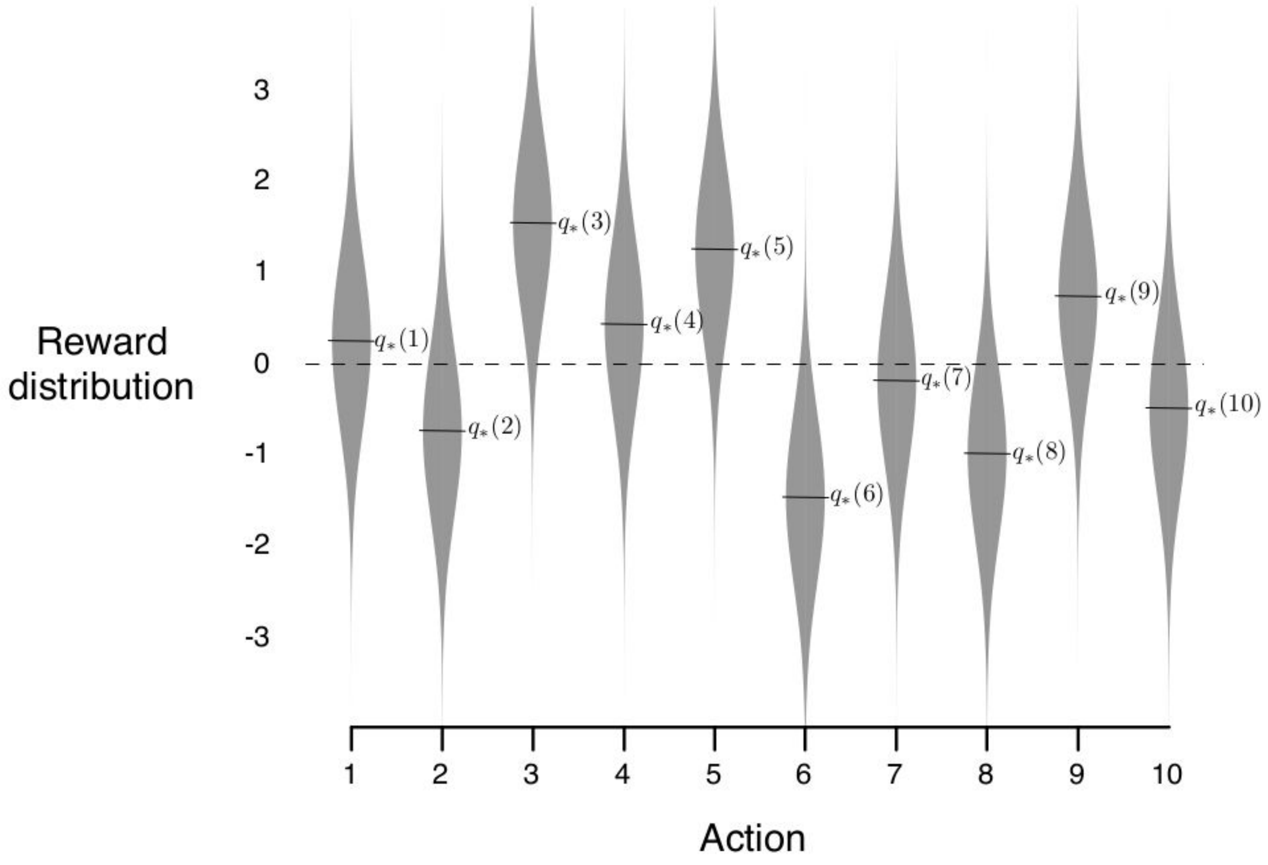
= 0.5 + 0.25

= 0.75

10-armed Testbed

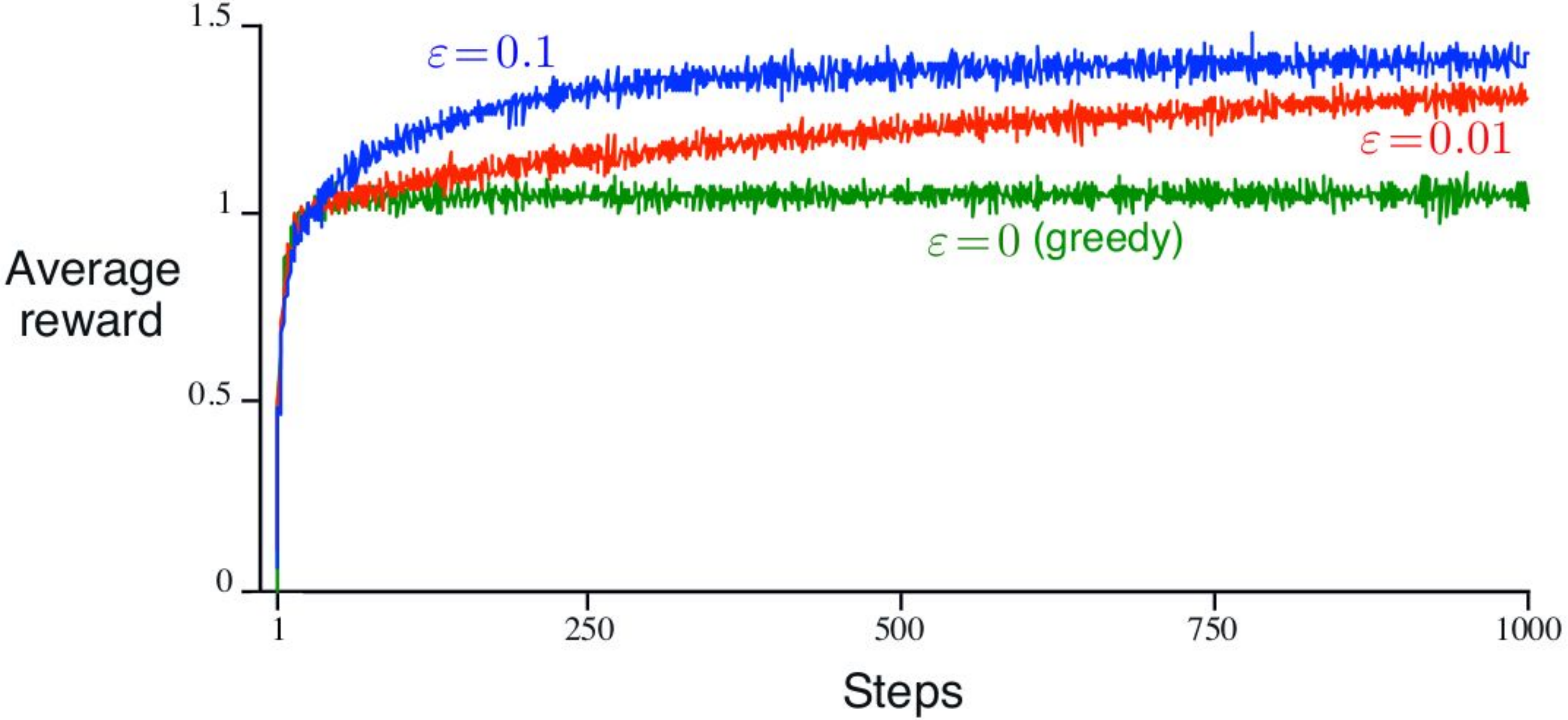
Example:

- A set of 2000 randomly generated k -armed bandit problems with k = 10
- Action values were selected according to a normal (Gaussian) distribution with mean 0 and variance 1.
- While selecting action A_t at time step t, the actual reward, R_t , was selected from a normal distribution with mean $q_*(At)$ and variance 1
- One Run : Apply a method for 1000 time steps to one of the bandit problems
- Perform 2000 runs, each run with a different bandit problem, to get an algorithms average behavior

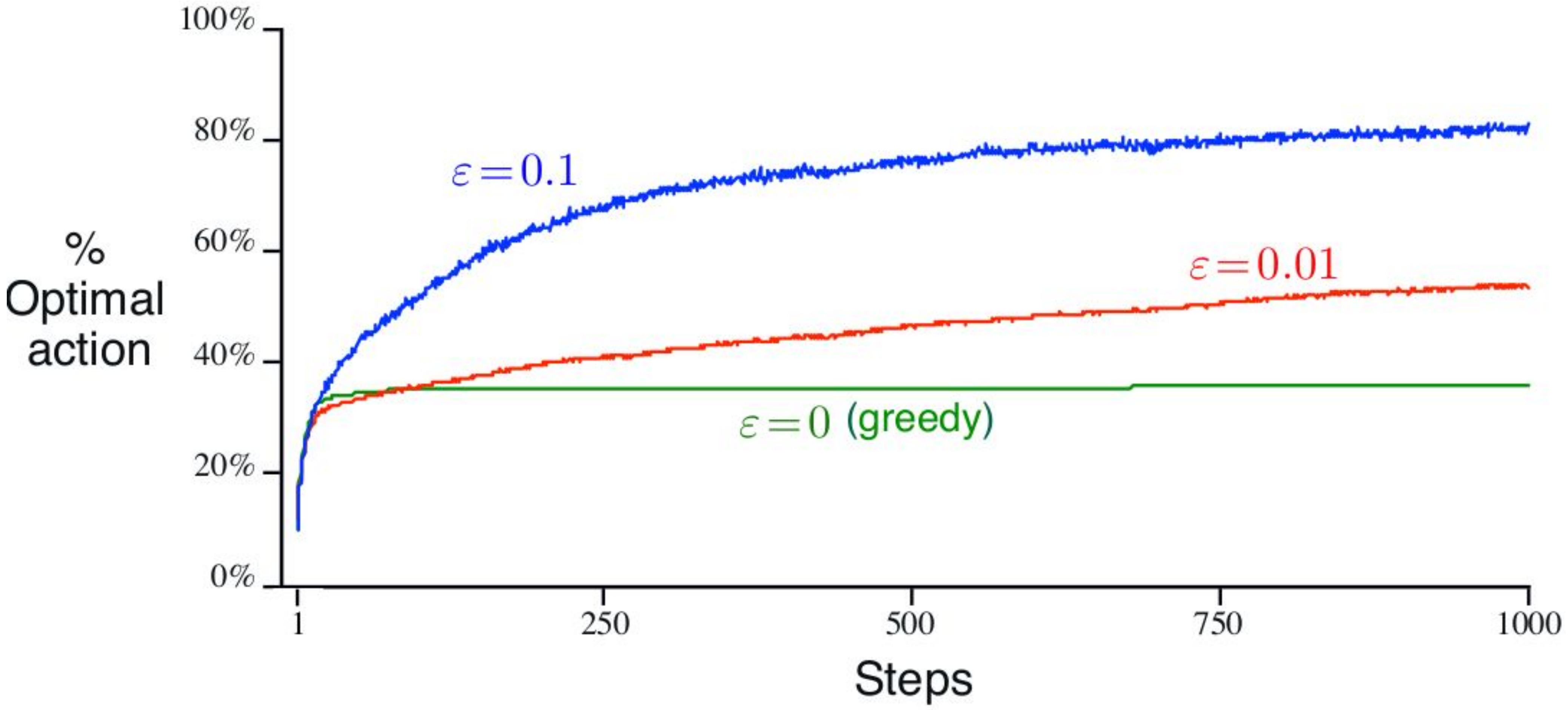


An example bandit problem from the 10-armed testbed

Average performance of ε-greedy action-value methods on the 10-armed testbed



Average performance of ε-greedy action-value methods on the 10-armed testbed



Discussion on Exploration vs. Exploitation

1) What if the reward variance is
    a. larger, say 10 instead of 1?
    b. zero ? [deterministic]

2) What if the bandit task is non-stationary? [that is, the true values of the actions changed over time]

Ex-2:

Consider a k -armed bandit problem with $k = 4$ actions, denoted 1, 2, 3, and 4.

Consider applying to this problem a bandit algorithm using ε-greedy action selection, sample-average action-value estimates, and initial estimates of $Q_1(a) = 0$, for all a.

Suppose the initial sequence of actions and rewards is $A_1 = 1, R_1 = 1, A_2 = 2, R_2 = 1, A_3 = 2, R_3 = 2, A_4 = 2, R_4 = 2, A_5 = 3, R_5 = 0$.

On some of these time steps the ε case may have occurred, causing an action to be selected at random.

On which time steps did this definitely occur? On which time steps could this possibly have occurred?

Incremental Implementation

- Efficient approach to compute the estimate of action-value;

$$Q_n \doteq \frac{R_1 + R_2 + \cdots + R_{n-1}}{n-1}.$$

- Given Qn and the nth reward, Rn , the new average of all n rewards can be computed as follows

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

Incremental Implementation

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

Note:

- StepSize decreases with each update
- We use α or $\alpha_t(a)$ to denote step size (constant / varies with each step)

Discussion:

Const vs. Variable step size?

$$
NewEstimate \leftarrow OldEstimate \; + \; StepSize \Big[Target - OldEstimate \Big]
$$

Bandit Algorithm with Incremental Update/ ε-greedy selection

Initialize, for $a = 1$ to k:

$$Q(a) \leftarrow 0$$
$$N(a) \leftarrow 0$$

Loop forever:

$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1 - \varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$
$$R \leftarrow bandit(A)$$
$$N(A) \leftarrow N(A) + 1$$
$$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\left[R - Q(A)\right]$$

Non-stationary Problem

- Most RL problems are non-stationary !
- Give more weight to recent rewards than to long-past rewards !!!

$$Q_{n+1} \doteq Q_n + \alpha \Big[R_n - Q_n \Big]$$

Non-stationary Problem

- Most RL problems are non-stationary !
- Give more weight to recent rewards than to long-past rewards !!!

$$Q_{n+1} \doteq Q_n + \alpha\Big[R_n - Q_n\Big]$$

Exponential recency-weighted average

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

Optimistic Initial Values

- All the above discussed methods are ***biased*** by their initial estimates
- For sample average method the bias disappears once all actions have been selected at least once
- For methods with constant α, the bias is permanent, though decreasing over time
- Initial action values can also be used as a simple way of encouraging exploration.
- In 10 armed testbed, set initial estimate to +5 rather than 0.

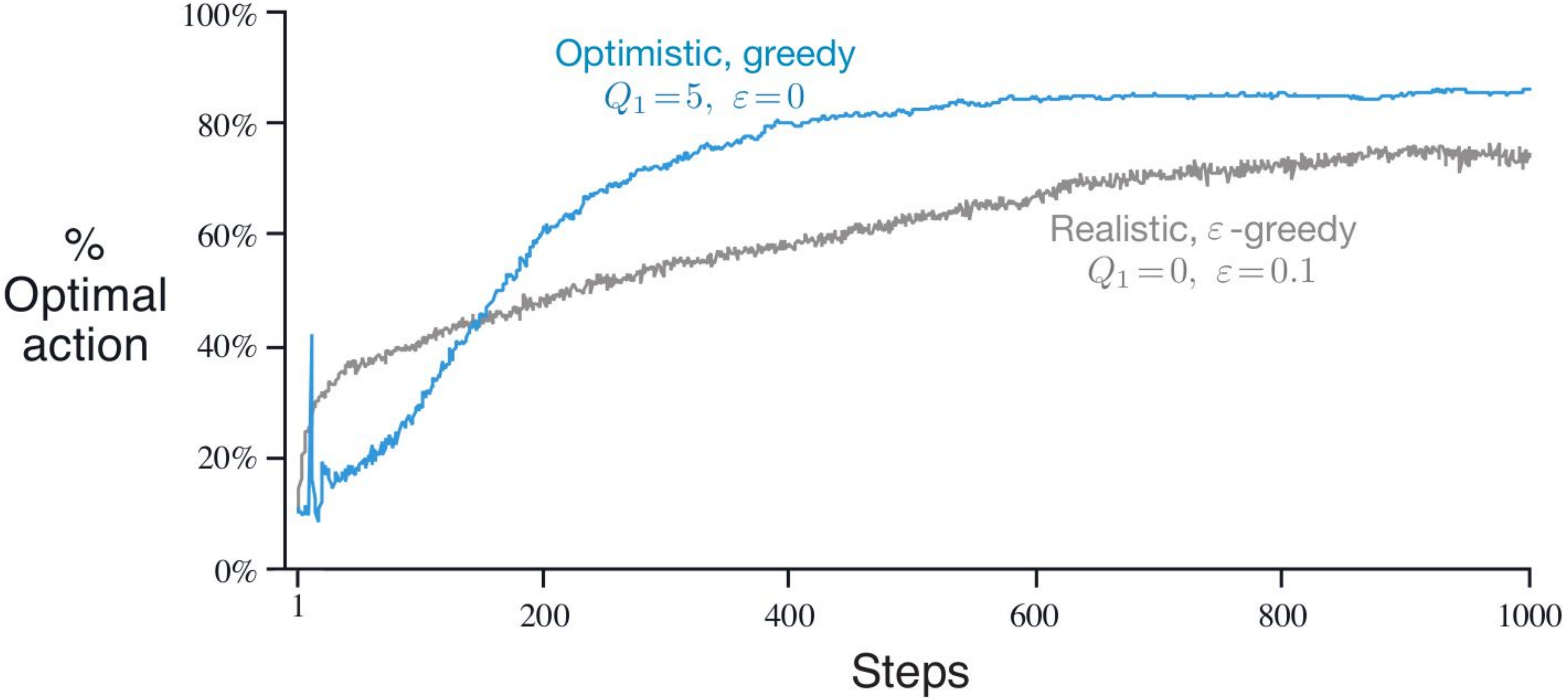
  This can encourage action-value methods to explore.

  Whichever actions are initially selected, the reward is less than the starting estimates;

  the learner switches to other actions, being disappointed with the rewards it is receiving.

  The result is that all actions are tried several times before the value estimates converge.

Optimistic Initial Values



Caution:
Optimistic Initial Values can only be considered as a simple trick that can be quite effective on stationary problems, but it is far from being a generally useful approach to encouraging exploration.

Question:
Explain how in the non-stationary scenario the optimistic initial values will fail (to explore adequately).

The effect of optimistic initial action-value estimates on the 10-armed testbed.
Both methods used a constant step-size parameter, $\alpha = 0.1$

Upper-Confidence-Bound Action Selection

- ε-greedy action selection forces the non-greedy actions to be tried,
  Indiscriminately, with no preference for those that are nearly greedy or particularly uncertain
- It would be better to select among the non-greedy actions according to their potential for actually being optimal
  Take into account both how close their estimates are to being maximal and the uncertainties in those estimates.

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

Upper-Confidence-Bound Action Selection

- Each time a is selected the uncertainty is presumably reduced
- Each time an action other than a is selected, t increases but $N_t(a)$ does not; because t appears in the numerator, the uncertainty estimate increases.
- Actions with lower value estimates, or that have already been selected frequently, will be selected with decreasing frequency over time

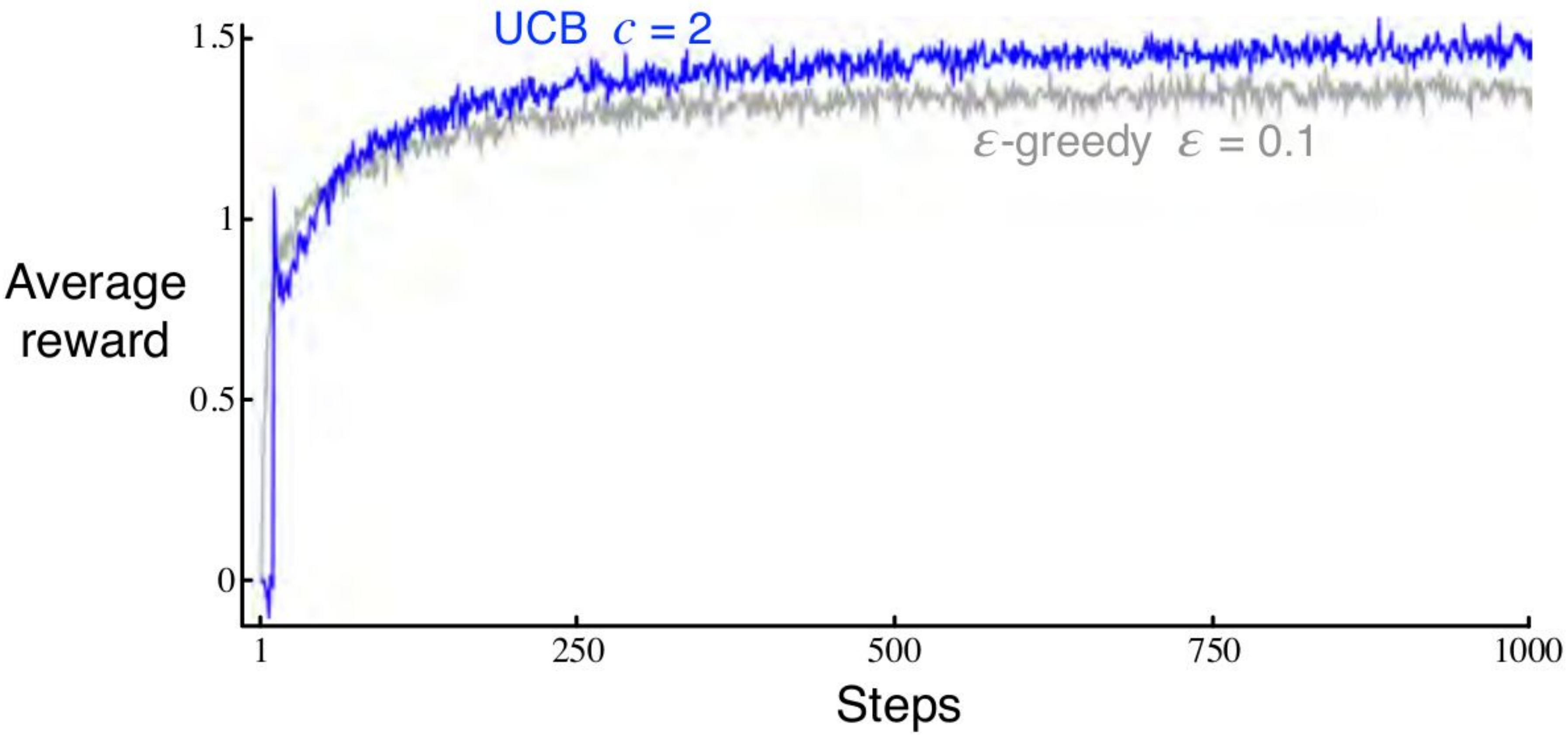
Action Value at time t for a

Confidence Level

Measure of Uncertainty

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

Upper-Confidence-Bound Action Selection



UCB often performs well, as shown here, but is more difficult than "-greedy to extend beyond bandits to the more general reinforcement learning settings

Policy-based algorithms

- Forget about action-value (Q) estimates, we don’t really care about them
- We care about what actions to chose

Let's assign a preference to each action and tweak its value

- Define $H_t(a)$ as a numerical preference value associated with action a
- Which action is selected?
  - $A_t = \underset{a}{\operatorname{argmax}}[H_t(a)]$
    - Hardmax results in no exploration -- deterministic action selection

Softmax!

Softmax function

- **Input**: vector of preferences
- **Output**: vector of probabilities forming a valid distribution
- $\Pr\{a_t = a\} = \frac{e^{H_t(a)}}{\sum_{a' \in A} e^{H_t(a')}} = \Pr(a)$
- $H_t\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{bmatrix} 6 \\ 9 \\ 2 \end{bmatrix}$
- $\text{softmax}\begin{pmatrix} 6 \\ 9 \\ 2 \end{pmatrix} = \begin{bmatrix} 0.047 \\ 0.952 \\ 0.00087 \end{bmatrix}$
- That is, with $\Pr(0.95)$ choose a_2, $\Pr(0.05)$ choose a_1, and $< \Pr(0.01)$ choose a_3

Softmax function

- Exploration – checked!
- Softmax provides another important attribute – **a differentiable policy**
- Say that we learn that action a_1 results in good relative performance
- Hardmax: $A_t = \underset{a}{\operatorname{argmax}}[H_t(a_1), H_t(a_2), H_t(a_3)]$
  - Change $H(a_1)$ such that $\Pr(a_1)$ is increased, $\frac{\partial \Pr(a_1)}{\partial H(a_1)} = NA$
- Softmax: $\Pr(a_1) = \frac{e^{H_t(a)}}{\sum_{a' \in A} e^{H_t(a')}}$
  - Change $H(a_1)$ such that $\Pr(a_1)$ is increased -> update towards $\frac{\partial \Pr(a_1)}{\partial H(a_1)}$

Gradient ascend

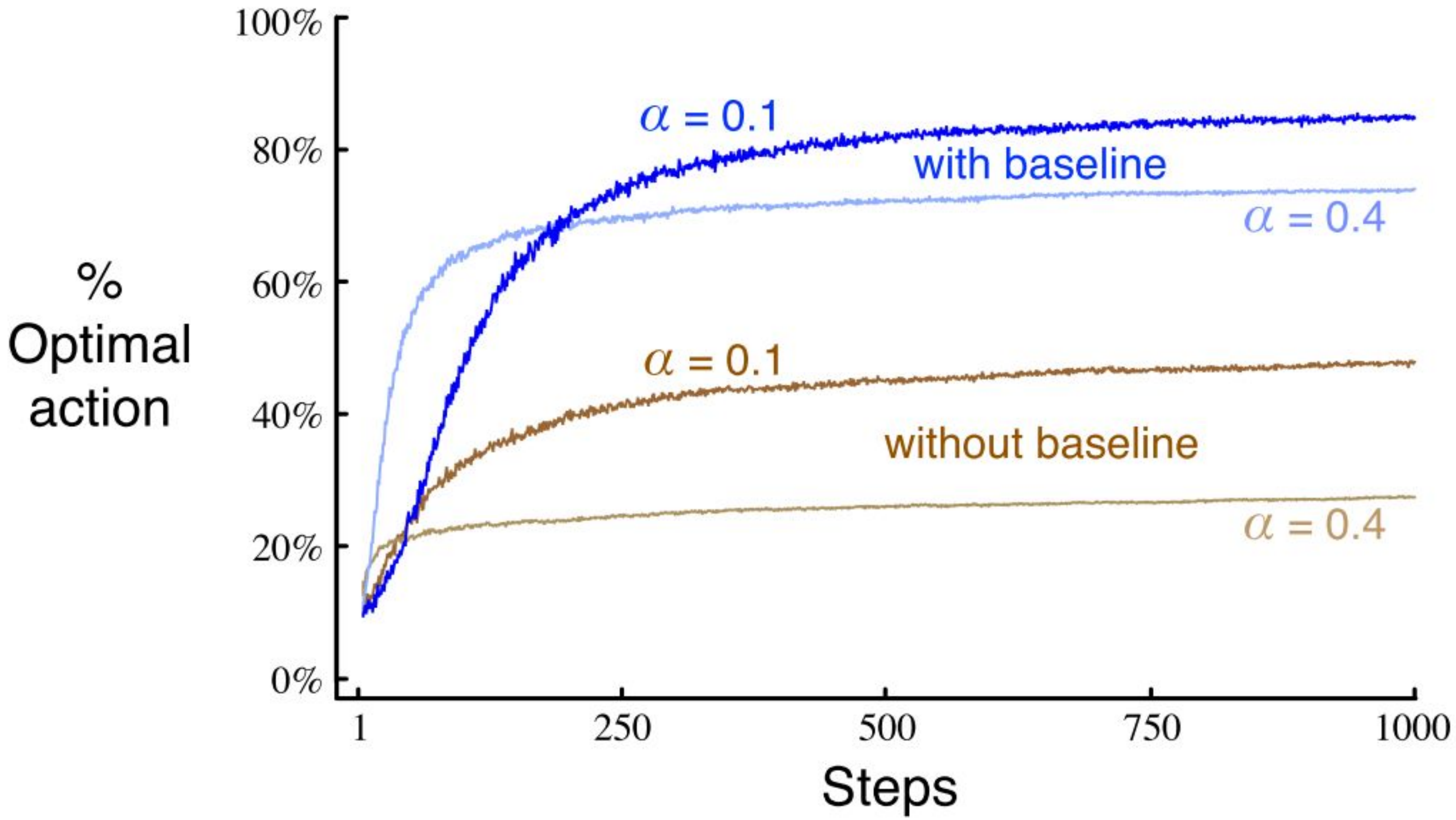
- $H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \overline{R_t})\boxed{\frac{\partial \Pr(A_t)}{\partial H(A_t)}}$?
- $\forall a \neq A_t, H_{t+1}(a) = H_t(a) + \alpha(R_t - \overline{R_t})\boxed{\frac{\partial \Pr(A_t)}{\partial H(a)}}$
- Update the preferences based on observed reward and a baseline reward ($\overline{R_t}$)
- If the observed reward is larger than the baseline:
  - Increase the preference of the chosen action, A_t
  - Decrease the preference of all other actions, $\forall a \neq A_t$
- Else do the opposite

Update Rule

On each step, after selecting action A t and receiving the reward Rt ,
Update the action preferences :

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \overline{R_t})(1 - \Pr(A_t))$$
$$\forall a \neq A_t, H_{t+1}(a) = H_t(a) - \alpha(R_t - \overline{R_t})\Pr(a)$$

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), \qquad \text{and}$$
$$H_{t+1}(a) \doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), \qquad \text{for all } a \neq A_t$$



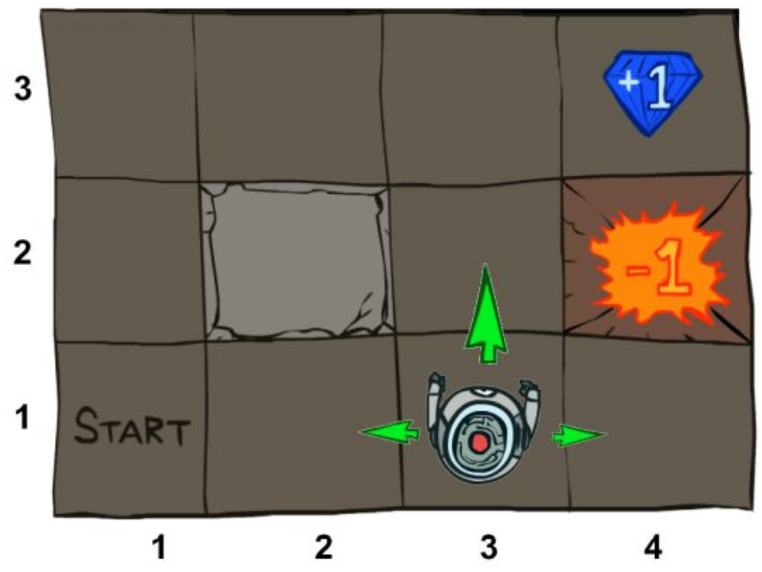
100%
80%
60%
40%
20%
0%
%
Optimal
action
α = 0.1
with baseline
α = 0.4
α = 0.1
without baseline
α = 0.4
1
250
500
750
1000
Steps

What did we learn?

- **Problem**: choose the action that results in highest expected reward
- **Assumptions**: 1. actions' expected reward is unknown, 2. we are confronted with the same problem over and over, 3. we are able to observe an action's outcome once chosen
- **Approach**: learn the actions' expected reward through exploration (value based) or learn a policy directly (policy based), exploit learnt knowledge to choose best action
- **Methods:** 1. greedy + initializing estimates optimistically, 2. epsilon-greedy, 3. Upper-Confidence-Bounds, 4. gradient ascend + soft-max

A different scenario

- Associative vs. Non-associative tasks ?
- Policy: A mapping from situations to the actions that are best in those situations
- (discuss) How do we extend the solution for non-associative task to an associative task?
    - **Approach**: Extend the solutions to non-stationary task to non-associative tasks
        - Works, if the true action values changes slowly
    - What if the context switching between the situations are made explicit?
        - How?
        - Need Special approaches !!!



Required Readings

1. Chapter-2 of Introduction to Reinforcement Learning, 2^{nd} Ed., Sutton & Barto
2. A Survey on Practical Applications of Multi-Armed and Contextual Bandits, Djallel Bouneffouf , Irina Rish [https://arxiv.org/pdf/1904.10040.pdf]

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Thank you !