8 - Multi-Armed Bandits & TD Learning

知识点 & 题目

Multi-Armed Bandits

An N-armed bandit is defined by a set of *random variables* $X_{i,k}$ where

- $1 \leq i \leq N$, such that i is the *arm* of the bandit; and
- k the index of the *play* of arm i.

Successive plays $X_{i,1}, X_{j,2}, X_{k,3} \ldots$ are assumed to be independently distributed according to an *unknown* law. That is, we do not know the probability distributions of the random variables.

Intuition: actions *a* applicable on *s* are the "arms of the bandit", and Q(s, a) corresponds to the random variables $X_{i,n}$.

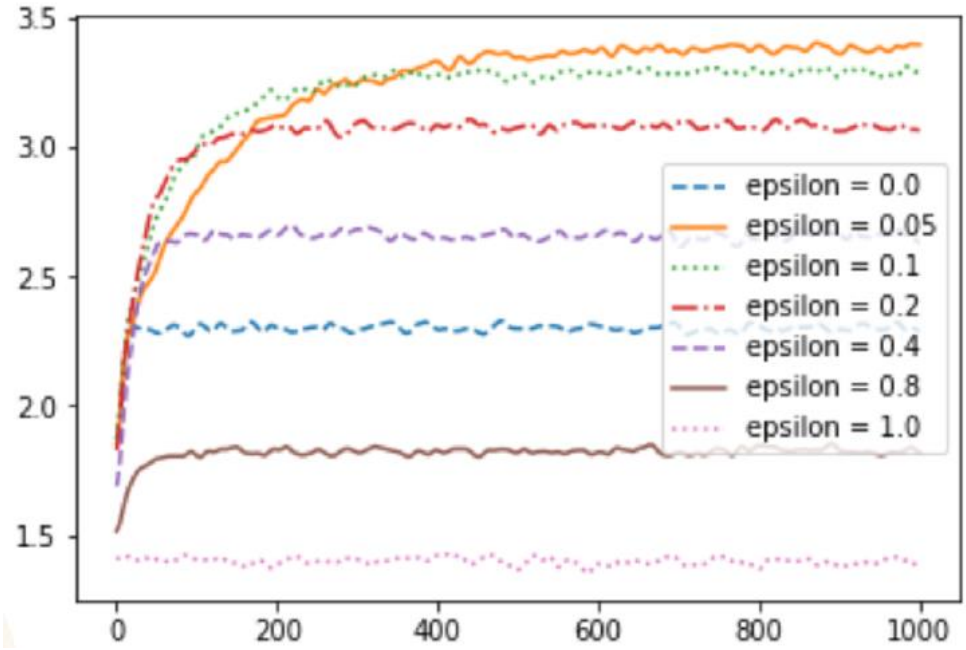
Regret: L8.1 P7

Exploration: e.g. select a random action

Exploitation: Use learned information to make selection

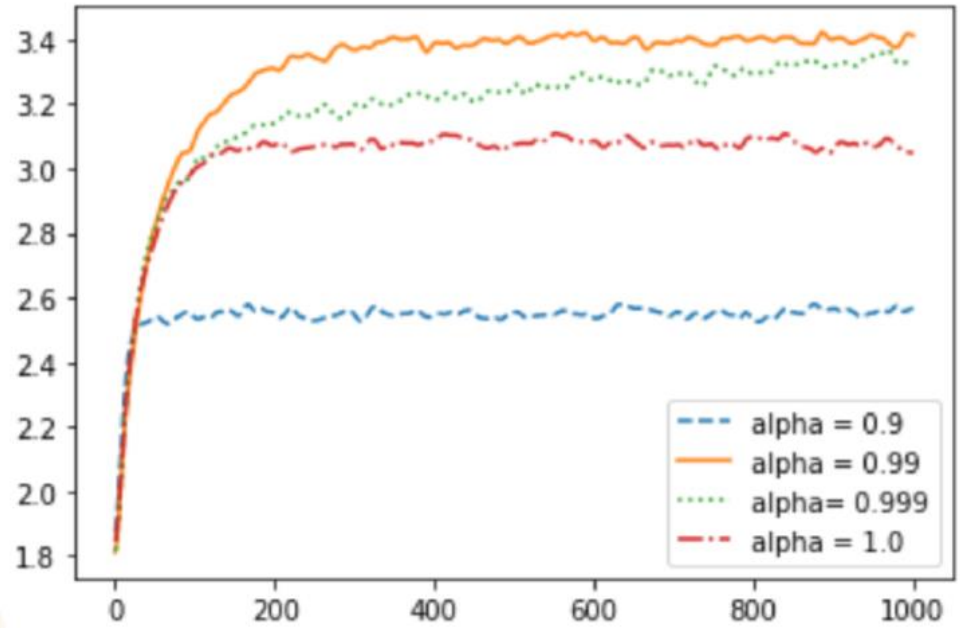
Epsilon greedy

ϵ**-greedy**: ϵ is a number in [0,1]. Each time we need to choose an arm, we choose a random arm with probability ϵ, and choose the arm with max $Q(s, a)$ with probability $1 - \epsilon$. Typically, values of ϵ around 0.05-0.1 work well.



Epsilon decreasing

ϵ-decreasing: The same as ϵ-greedy, ϵ decreases over time. A parameter α between [0,1] specifies the *decay*, such that $\epsilon := \epsilon.\alpha$ after each action is chosen.

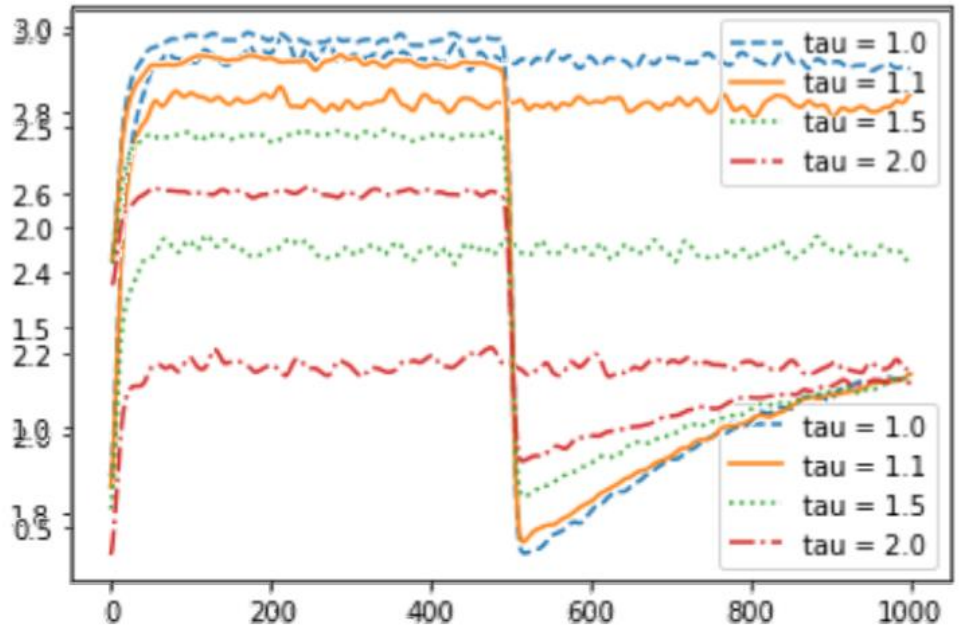


- increase alpha -> reduce number of explorations
- Gain more info -> explore less / exploit more

Softmax: This is *probability matching strategy*, which means that the probability of each action being chosen is dependent on its Q-value so far. Formally:

$$\frac{e^{Q(s,a)/\tau}}{\sum_{b=1}^{n} e^{Q(s,b)/\tau}}$$

in which τ is the *temperature*, a positive number that dictates how much of an influence the past data has on the decision.



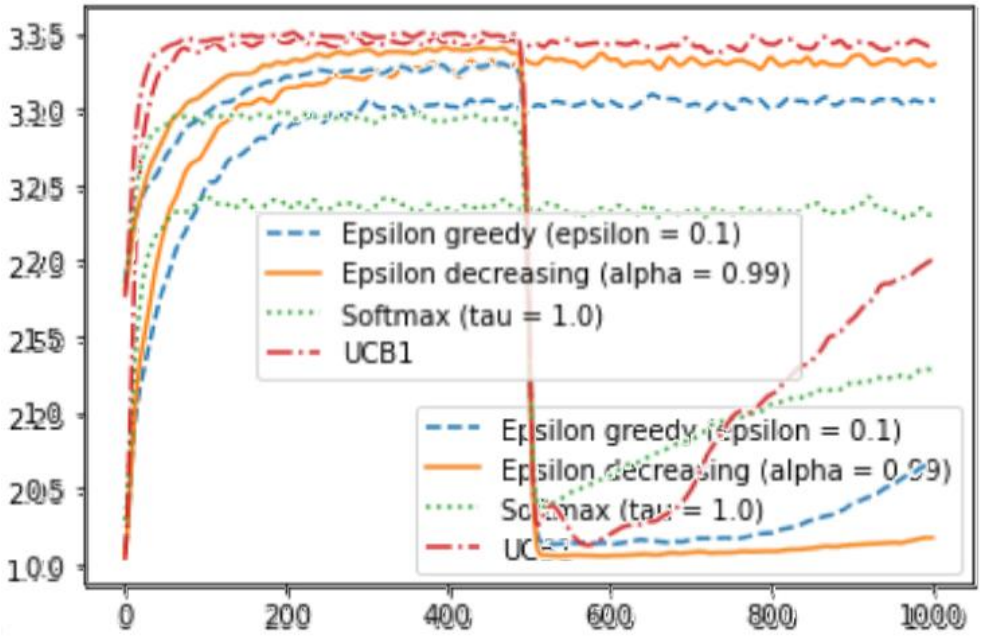
- At step 500, probabilities are changed and old information are not useful.
- Tao =1.0 recovers the fastest because more likely to sample actions previously bad but now good.

Upper Confidence Bounds

UCB1 policy $\pi(s)$

$$\pi(s) := \underset{a \in A(s)}{\operatorname{argmax}} \left(Q(s,a) + \sqrt{\frac{2 \ln N(s)}{N(s,a)}} \right)$$

$Q(s,a)$ is the estimated Q-value.
$N(s)$ is the number of times s has been visited.
$N(s,a)$ is the number of times times a has been executed in s.



- Upper Confidence Bounds (UCB1) for Multi-Armed Bandits makes a good selection policy.
- UCB1 (with slight modifications) balances exploitation and exploration remarkable well.
- The Fear Of Missing Out is an excellent motivator for exploration.

TD Learning

$$Q(s,a) \leftarrow \underbrace{Q(s,a)}_{\text{old value}} + \overbrace{\alpha}^{\text{learning rate}} \cdot [\overbrace{r}^{\text{reward}} + \overbrace{\gamma}^{\text{discount factor}} \cdot V(s') - \overbrace{Q(s,a)}^{\text{do not count extra } Q(s,a)}]$$

$\underbrace{\qquad}_{\text{TD target}}$

- alpha = [0, 1]
- [part] -> TD estimate
- V(s') = $\max_{a'} Q(s', a')$

Off-Policy TD: Q-Learning EX L8.2 P9

Input: MDP $M = \langle S, s_0, A, P_a(s' \mid s), r(s, a, s') \rangle$
Output: Q-function Q

Initialise Q arbitrarily; e.g., $Q(s, a) = 0$ for all s and a

Repeat (for each episode)
    $s \leftarrow$ the first state in episode e
    Repeat (for each step in episode e)
        Select action a to apply in s; e.g. Q and a multi-armed bandit algorithm
        Execute action a in state s
        Observe reward r and new state s'
        $\delta \leftarrow r + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a)$
        $Q(s, a) \leftarrow Q(s, a) + \alpha \cdot \delta$
        $s \leftarrow s'$
    Until s is the last state of episode e (a terminal state)

On-Policy TD: SARSA

Input: MDP $M = \langle S, s_0, A, P_a(s' \mid s), r(s, a, s') \rangle$
Output: Q-function Q

Initialise Q arbitrarily; e.g., $Q(s, a) = 0$ for all s and a

Repeat (for each episode)
    $s \leftarrow$ the first state in episode e
    Select action a to apply in s using Q and a multi-armed bandit algorithm
    Repeat (for each step in episode e)
        Execute action a in state s
        Observe reward r and new state s'
        Select action a' to apply in s' using Q and a multi-armed bandit algorithm
        $\delta \leftarrow r + \gamma \cdot Q(s', a') - Q(s, a)$
        $Q(s, a) \leftarrow Q(s, a) + \alpha \cdot \delta$
        $s \leftarrow s'$
        $a \leftarrow a'$
    Until s is the last state of episode e (a terminal state)

SARSA vs. Q-Learning

$$\delta \leftarrow r + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a)$$

$$\delta \leftarrow r + \gamma \cdot Q(s', a') - Q(s, a)$$

- **Off-Policy: Prior experience**
- **On-Policy: Learning on the job**
- Limitations
    - Finite number of states and actions
    - Unavailable on continuous actions

- Q-table requires huge memory

题目

Quiz

Question 1

3 / 3 pts

Match the following definitions of to names of multi-armed bandit algorithms

Exploit best action with probability 1-epsilon and random from all other actions with epsilon probability	epsilon-greedy
Exploit actions proportionally based on their Q-value	softmax
Exploit Q-value and exploit based on number of times an option has been chosen	UCB

Other Incorrect Match Options:

- epsilon-decreasing

Question 2

0 / 1 pts

Model-free reinforcement learning is:

- Learning the environment from experience and rewards so we can construct a policy
- Q-learning but not SARSA
- Learning a policy directly from experience and rewards
- Any learning without a model

Question 3

1 / 1 pts

Which of the following is not true?

- SARSA is on-policy learning
- SARSA is model-free but can be used with a model
- Q-learning learns Q values but SARSA does not

  Both learn Q values, but the letter 'Q' is only in Q-learning for no good scientific reason :)
- Q-learning is model-free learning

Question 4

1 / 1 pts

SARSA is an on-policy learning approach because:

- (●) It updates the Q values based on the actual policy followed

  Correct! Because the updates follow the policy, it is "on policy"; whereas Q-learning updates based on maximum expected future reward, which is not what the policy will follow.

- ( ) It updates the policy directly

- ( ) It updates based on the maximum estimated future reward, which is what we follow on the policy

Question 5

2 / 2 pts

State	Action			
	North	South	East	West
(0,0)	0.53	0.36	0.36	0.21
(0,1)	0.61	0.27	0.23	0.23
...				
(2,2)	0.79	0.72	0.90	0.72
(2,3)	0.90	0.78	0.99	0.81

Assume a learning rate $\alpha = 0.1$ and discount factor $\gamma = 0.9$, use Q-learning to update the Q value if the action North is chosen in state (0,0), receiving no reward, and transitioning to state (0,1)

0.5319

0.5319 (with margin: 0.0019)

$$
\begin{aligned}
Q((0,0),N) &\leftarrow Q((0,0),N) + \alpha[r + \gamma \max_{a'} Q((0,0),a') - Q((0,0),N)] \\
&\leftarrow 0.53 + 0.1[0 + 0.9 \cdot Q((0,1),N) - Q((0,0),N)] \\
&\leftarrow 0.53 + 0.1[0 + 0.9 \cdot 0.61 - 0.53]
\end{aligned}
$$

$$\leftarrow \quad 0.5319$$

Assume a learning rate $\alpha = 0.2$ and discount factor $\gamma = 0.9$, use SARSA to update the Q value if the action North is chosen in state (0,0), receiving no reward, and transitioning to state (0,1), and action South being chosen from state (0,1).

Note the learning rate parameter is 0.2, not 0.1 as in the previous example.

0.4726

0.4726 (with margin: 0.0026)

$$\begin{aligned} Q((0,0),N) &\leftarrow Q((0,0),N) + \alpha[r + \gamma Q((0,1),S) - Q((0,0),N)] \\ &\leftarrow 0.53 + 0.2[0 + 0.9 \cdot Q((0,1),S) - Q((0,0),N)] \\ &\leftarrow 0.53 + 0.2[0 + 0.9 \cdot 0.27 - 0.53] \\ &\leftarrow 0.4726 \end{aligned}$$