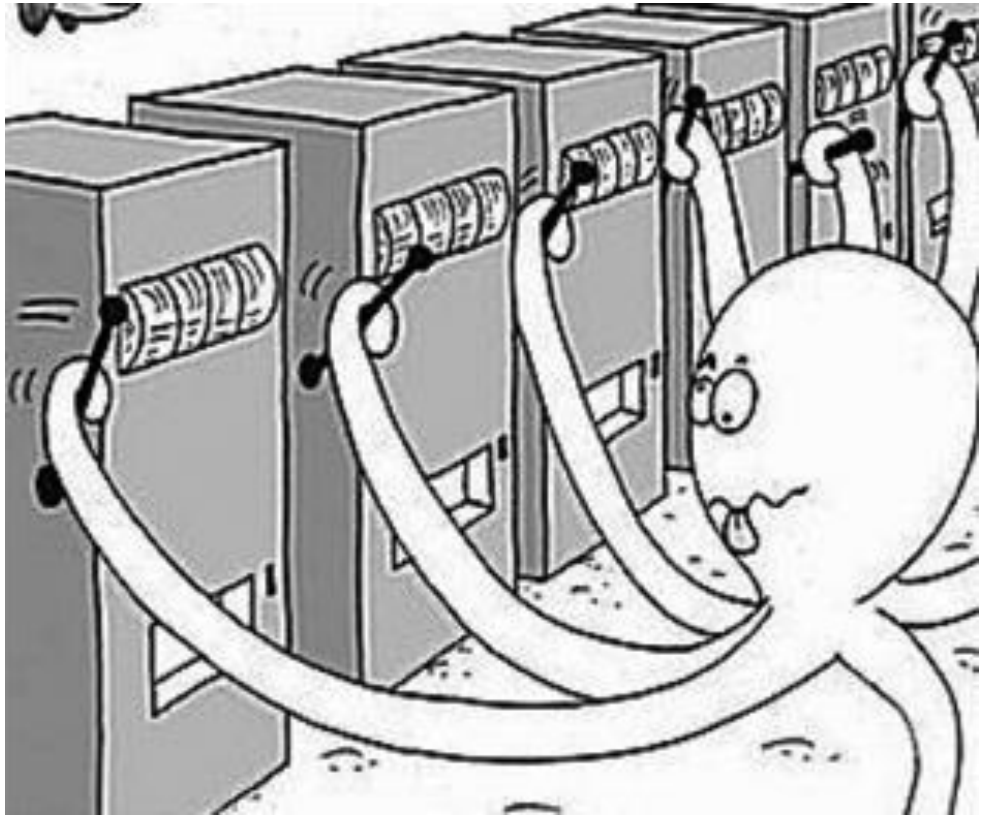
Multi-Armed Bandits

ChengCheng Han

March 6， 2020

一、 Introduction

- **Multiple arms.**
- When pulled, an arm produces a random payout drawn **independently of the past.**
- **the distribution** of payouts corresponding to each arm **is not listed.**
- **maximizes the cumulative payout earned.**

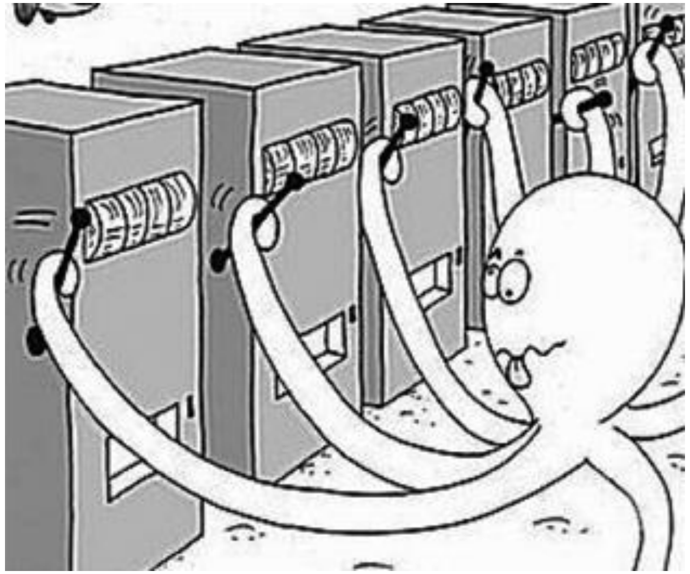


一、 Introduction

DaSE Data Science & Engineering

Exploitation Make the best decision given current information.

Exploration Gather more information



一、 Introduction

Restaurant Selection

Exploitation Go to your favourite restaurant

Exploration Try a new restaurant

Online Advertisements

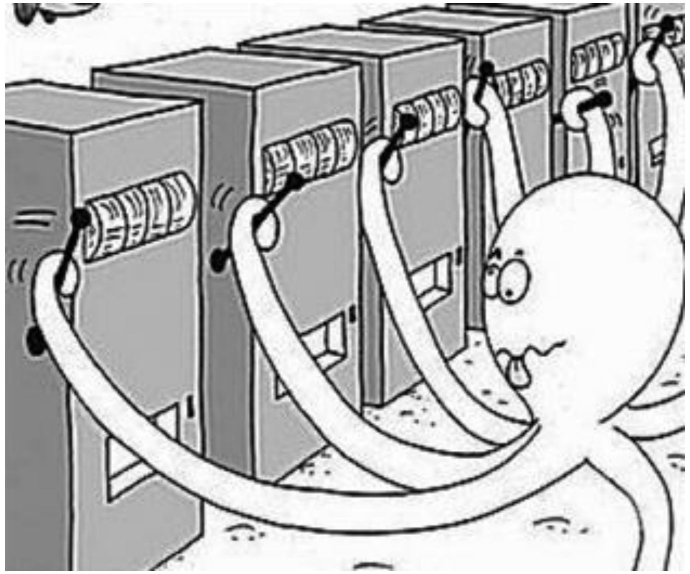
Exploitation Show the most successful advert

Exploration Show a different advert

Oil Drilling

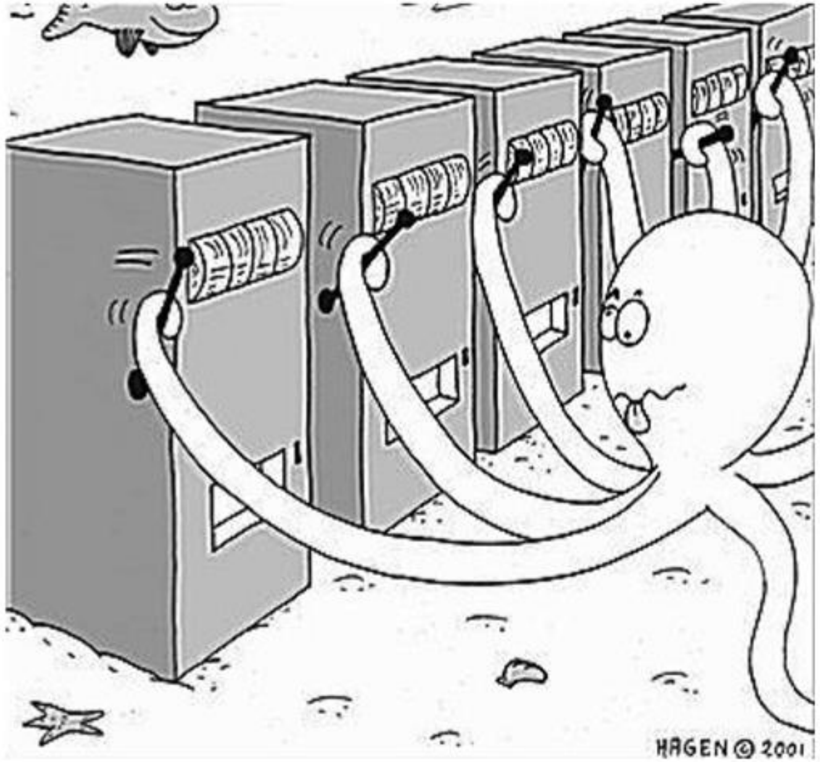
Exploitation Drill at the best known location

Exploration Drill at a new location



一、 Introduction

- A multi-armed bandit is a tuple $\langle \mathcal{A}, \mathcal{R} \rangle$
- $\mathcal{A}$ is a known set of m actions (or "arms")
- $\mathcal{R}^a(r) = \mathbb{P}[r|a]$ is an unknown probability distribution over rewards
- At each step t the agent selects an action $a_t \in \mathcal{A}$
- The environment generates a reward $r_t \sim \mathcal{R}^{a_t}$
- The goal is to maximise cumulative reward $\sum_{\tau=1}^{t} r_\tau$



一、 Introduction

- The *action-value* is the mean reward for action a,

$$Q(a) = \mathbb{E}\left[r|a\right]$$

- The *optimal value* V^* is

$$V^* = Q(a^*) = \max_{a \in \mathcal{A}} Q(a)$$

DaSE Data Science & Engineering

Regret

- The *regret* is the opportunity loss for one step

$$l_t = \mathbb{E}\left[V^* - Q(a_t)\right]$$

- The *total regret* is the total opportunity loss

$$L_t = \mathbb{E}\left[\sum_{\tau=1}^{t} V^* - Q(a_\tau)\right]$$

- Maximise cumulative reward $\equiv$ minimise total regret

一、 Introduction

DaSE Data Science & Engineering

Total Regret

$$
\begin{aligned}
L_t &= \mathbb{E}\left[\sum_{\tau=1}^{t} V^* - Q(a_\tau)\right] \\
&= \sum_{a \in \mathcal{A}} \mathbb{E}\left[N_t(a)\right]\left(V^* - Q(a)\right) \\
&= \sum_{a \in \mathcal{A}} \mathbb{E}\left[N_t(a)\right]\Delta_a
\end{aligned}
$$

- The *count* $N_t(a)$ is expected number of selections for action a
- The *gap* Δ_a is the difference in value between action a and optimal action a^*, $\Delta_a = V^* - Q(a)$

一、 Introduction

- A good algorithm ensures small counts for large gaps
- Problem: gaps are not known!

$$
\begin{aligned}
L_t &= \mathbb{E}\left[\sum_{\tau=1}^{t} V^* - Q(a_\tau)\right] \\
&= \sum_{a \in \mathcal{A}} \mathbb{E}\left[N_t(a)\right]\left(V^* - Q(a)\right) \\
&= \sum_{a \in \mathcal{A}} \mathbb{E}\left[N_t(a)\right]\Delta_a
\end{aligned}
$$

二、 Algorithm

- We consider algorithms that estimate $\hat{Q}_t(a) \approx Q(a)$
- Estimate the value of each action by Monte-Carlo evaluation

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{t=1}^{T} r_t \mathbf{1}(a_t = a)$$

DaSE Data Science & Engineering

1. *Greedy Algorithm*

- The *greedy* algorithm selects action with highest value

$$a_t^* = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \hat{Q}_t(a)$$

- Greedy can lock onto a suboptimal action forever
- $\Rightarrow$ Greedy has linear total regret

二、 Algorithm

2. $\epsilon - Greedy\ Algorithm$

- The ϵ-*greedy* algorithm continues to explore forever
  - With probability $1-\epsilon$ select $a = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ \hat{Q}(a)$
  - With probability ϵ select a random action
- Constant ϵ ensures minimum regret

$$l_t \geq \frac{\epsilon}{\mathcal{A}} \sum_{a \in \mathcal{A}} \Delta_a$$

- $\Rightarrow$ ϵ-greedy has linear total regret

二、 Algorithm

2. $\epsilon - Greedy\ Algorithm$

- The ϵ-*greedy* algorithm continues to explore forever
  - With probability $1 - \epsilon$ select $a = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \hat{Q}(a)$
  - With probability ϵ select a random action
- Constant ϵ ensures minimum regret

$$l_t \geq \frac{\epsilon}{\mathcal{A}} \sum_{a \in \mathcal{A}} \Delta_a$$

- $\Rightarrow$ ϵ-greedy has linear total regret

二、 Algorithm

Optimistic Initialization

- Simple and practical idea: initialise $Q(a)$ to high value
- Update action value by incremental Monte-Carlo evaluation
- Starting with $N(a) > 0$

$$\hat{Q}_t(a_t) = \hat{Q}_{t-1} + \frac{1}{N_t(a_t)}(r_t - \hat{Q}_{t-1})$$

二、 Algorithm

Optimistic Initialization

$$\hat{Q}_t(a_t) = \hat{Q}_{t-1} + \frac{1}{N_t(a_t)}(r_t - \hat{Q}_{t-1})$$

- Encourages systematic exploration early on
- But can still lock onto suboptimal action
- $\Rightarrow$ greedy $+$ optimistic initialisation has linear total regret
- $\Rightarrow$ ϵ-greedy $+$ optimistic initialisation has linear total regret

二、 Algorithm

3. $\boldsymbol{Delay\ \epsilon_t - greedy\ Algorithm}$

- Pick a decay schedule for $\epsilon_1, \epsilon_2, \ldots$
- Consider the following schedule

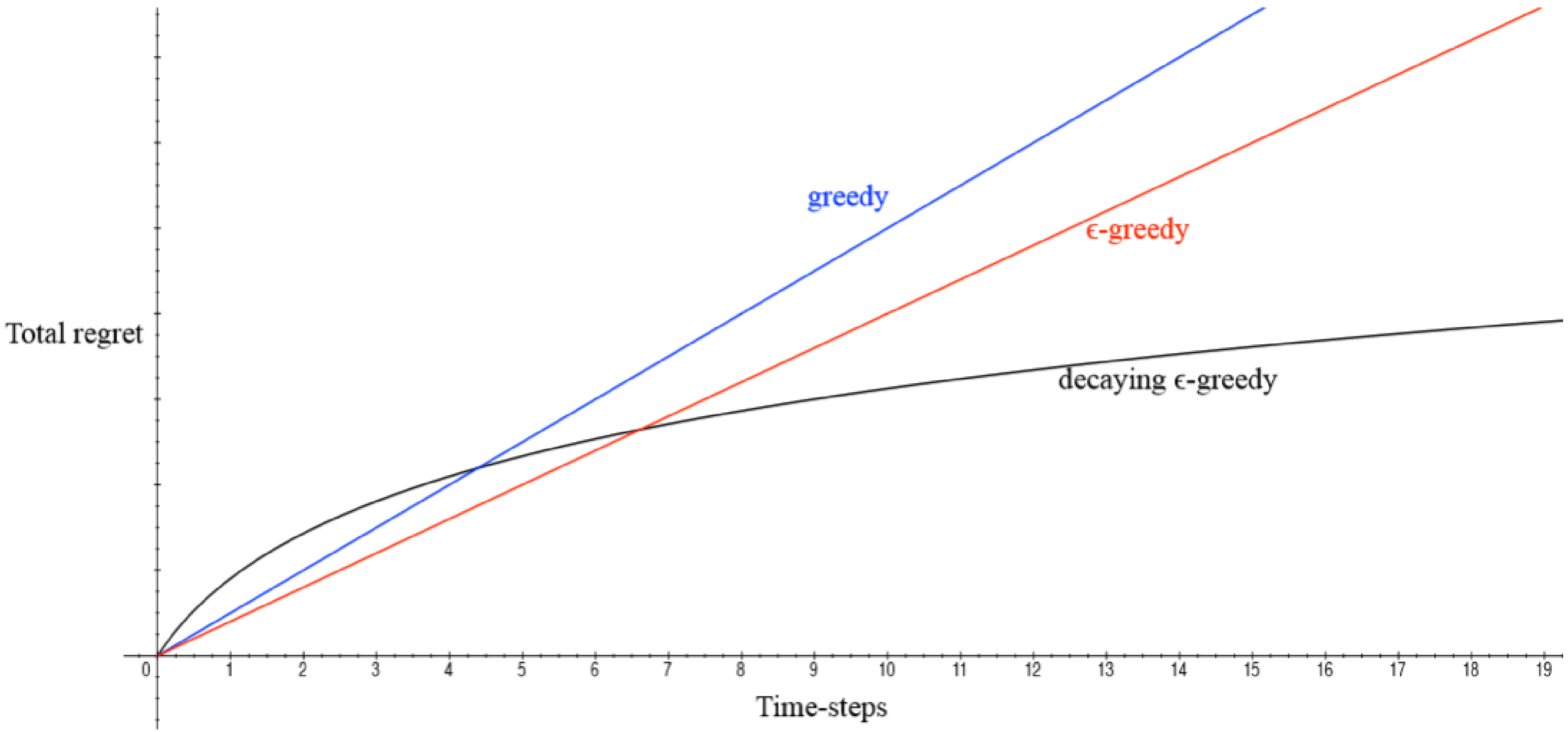
$$c > 0$$

$$d = \min_{a|\Delta_a > 0} \Delta_i$$

$$\epsilon_t = \min \left\{ 1, \frac{c|\mathcal{A}|}{d^2 t} \right\}$$

二、 Algorithm

3. $Delay\ \epsilon_t - greedy\ Algorithm$



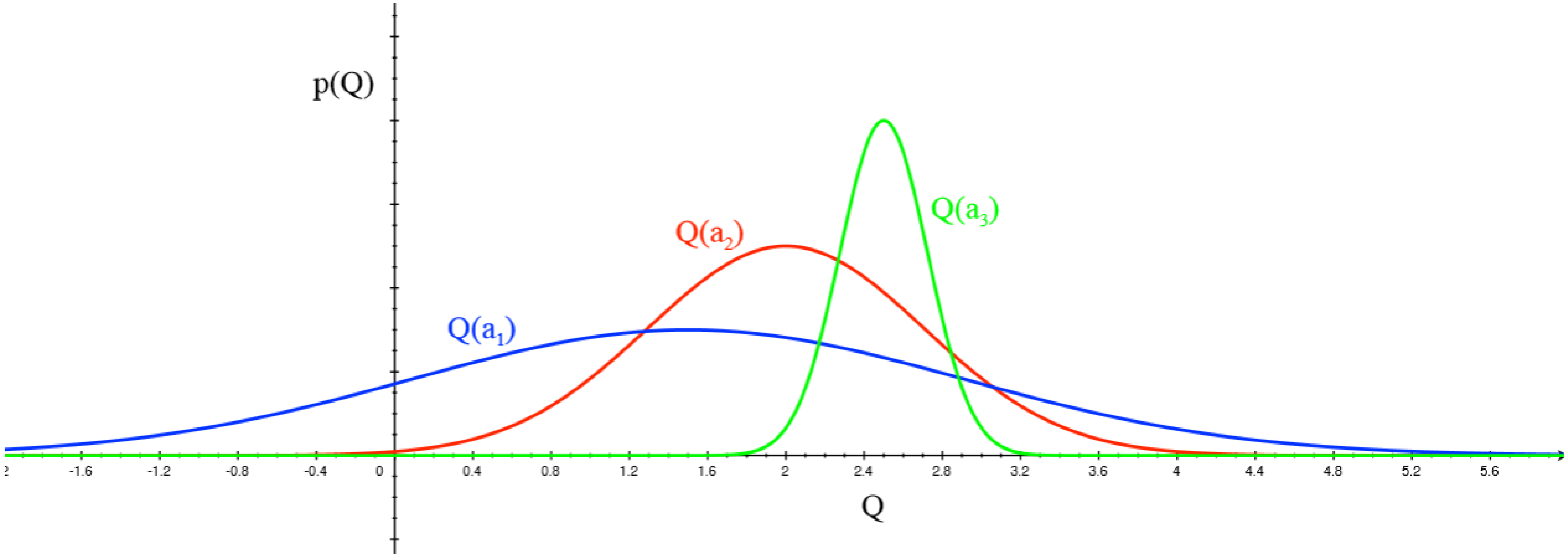
3. *Delay* ϵ_t − *greedy Algorithm*

- Decaying ϵ_t-greedy has *logarithmic* asymptotic total regret!
- Unfortunately, schedule requires advance knowledge of gaps
- Goal: find an algorithm with sublinear regret for any multi-armed bandit (without knowledge of $\mathcal{R}$)

二、 Algorithm

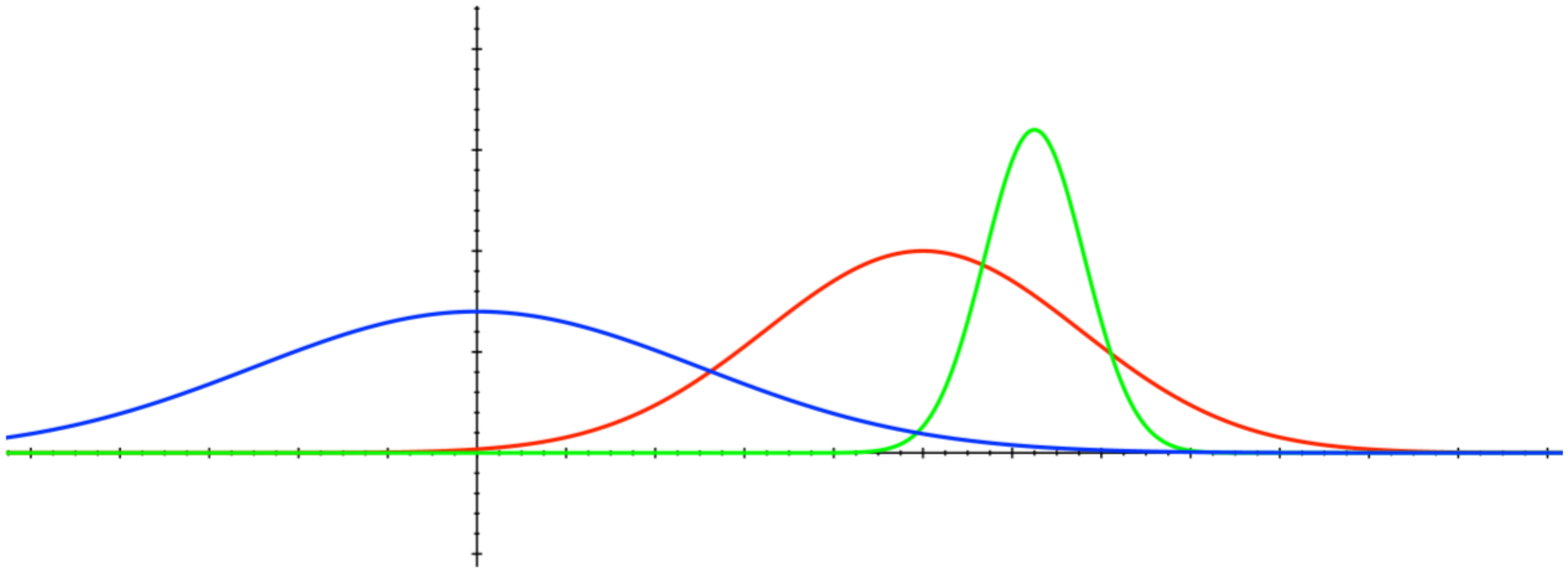
DaSE Data Science & Engineering

Optimism in the Face of Uncertainty
(不确定行为优先探索)



二、 Algorithm

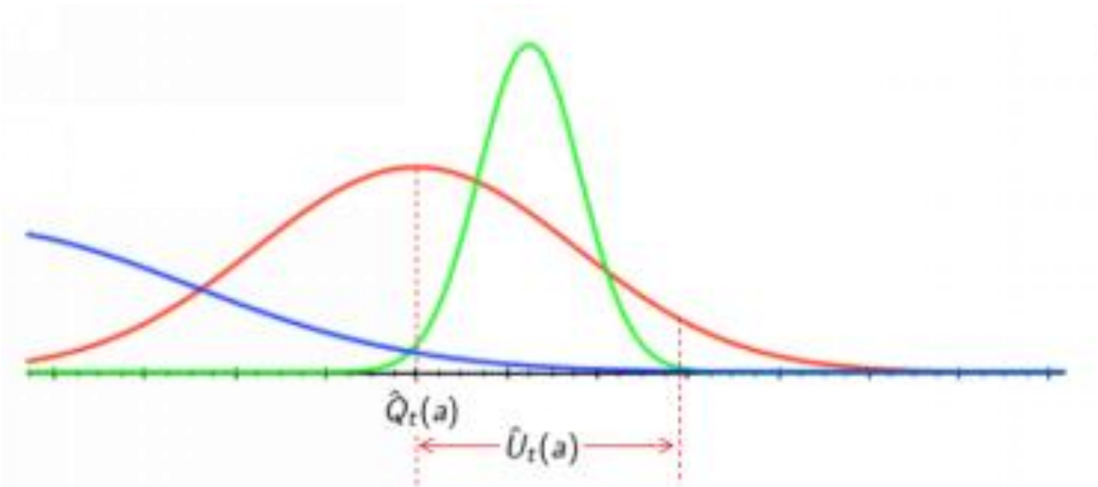
Optimism in the Face of Uncertainty
(不确定行为优先探索)



- After picking blue action

DaSE Data Science & Engineering

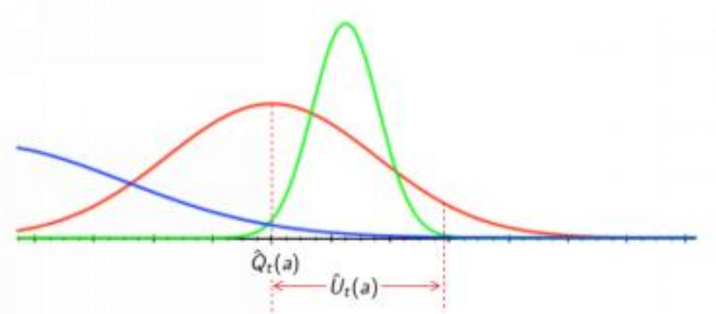
4. *Upper Cofidence Bounds(UCB)*



- Estimate an upper confidence $\hat{U}_t(a)$ for each action value
- Such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with high probability

二、 Algorithm

4. *Upper Cofidence Bounds(UCB)*



- This depends on the number of times $N(a)$ has been selected
  - Small $N_t(a) \Rightarrow$ large $\hat{U}_t(a)$ (estimated value is uncertain)
  - Large $N_t(a) \Rightarrow$ small $\hat{U}_t(a)$ (estimated value is accurate)
- Select action maximising Upper Confidence Bound (UCB)

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ \hat{Q}_t(a) + \hat{U}_t(a)$$

二、 Algorithm

4. *Upper Cofidence Bounds(UCB)*

Theorem (Hoeffding's Inequality)

Let $X_1, \ldots, X_t$ *be i.i.d. random variables in [0,1], and let* $\overline{X}_t = \frac{1}{\tau}\sum_{\tau=1}^{t} X_\tau$ *be the sample mean. Then*

$$\mathbb{P}\left[\mathbb{E}\left[X\right] > \overline{X}_t + u\right] \le e^{-2tu^2}$$

$$\Downarrow$$

$$\mathbb{P}\left[Q(a) > \hat{Q}_t(a) + U_t(a)\right] \le e^{-2N_t(a)U_t(a)^2}$$

DaSE Data Science & Engineering

4. *Upper Cofidence Bounds(UCB)*

$$\mathbb{P}\left[Q(a) > \hat{Q}_t(a) + U_t(a)\right] \leq e^{-2N_t(a)U_t(a)^2}$$

$$e^{-2N_t(a)U_t(a)^2} = p$$

$$U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}} \xrightarrow{p = t^{-4}} U_t(a) = \sqrt{\frac{2\log t}{N_t(a)}}$$

$$\Downarrow$$

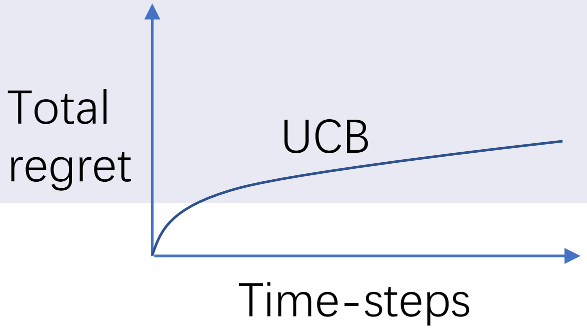
$$a_t = \underset{a\in\mathcal{A}}{\operatorname{argmax}}\ Q(a) + \sqrt{\frac{2\log t}{N_t(a)}}$$

二、 Algorithm

4. *Upper Cofidence Bounds(UCB)*

The UCB algorithm achieves logarithmic asymptotic total regret

$$\lim_{t\to\infty} L_t \leq 8 \log t \sum_{a|\Delta_a>0} \Delta_a$$

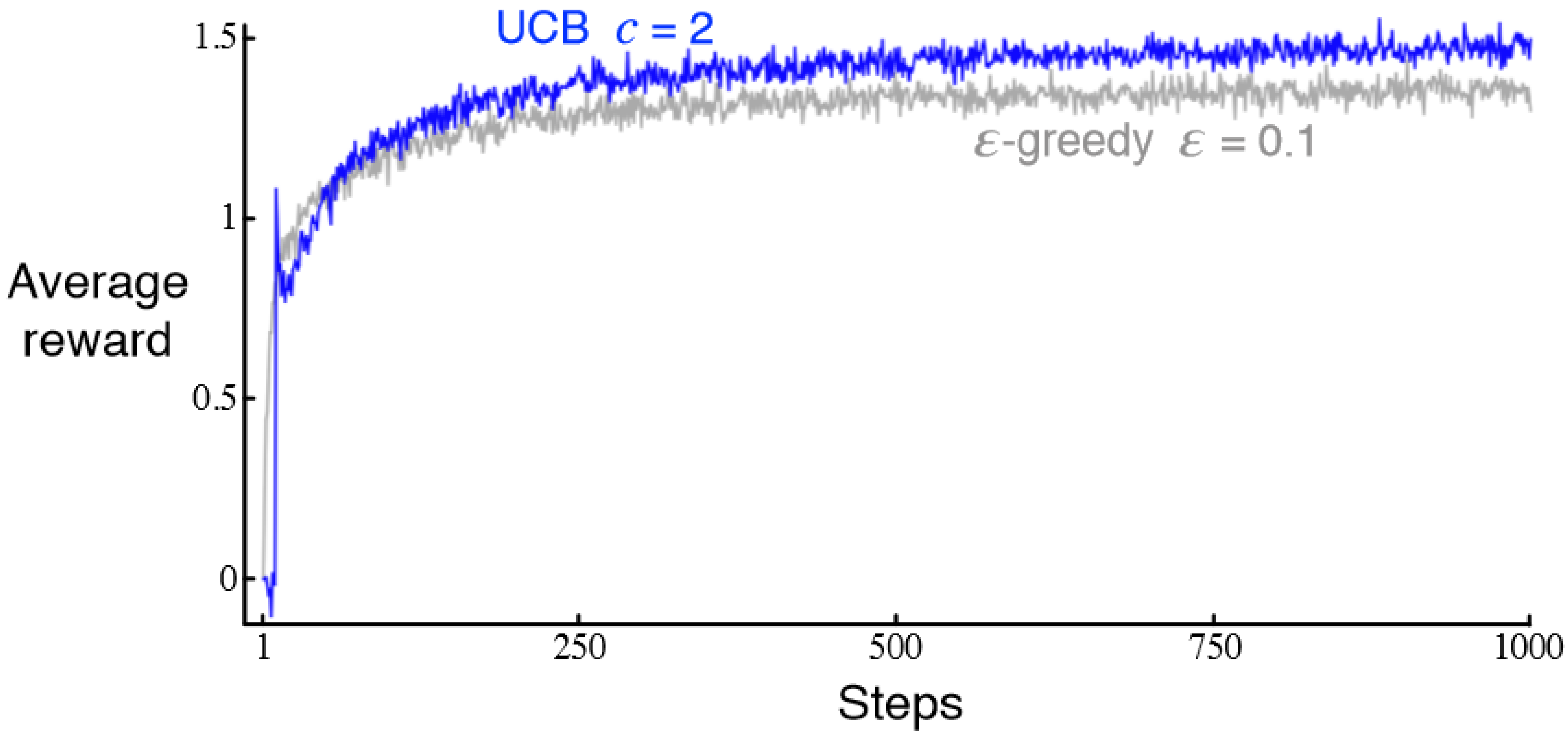


Finite-time Analysis of the Multiarmed Bandit Problem

二、 Algorithm

4. *Upper Cofidence Bounds(UCB)*

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



DaSE Data Science & Engineering

Frequentist vs Bayesian

Frequentist

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{t=1}^{T} r_t \mathbf{1}(a_t = a)$$

$$t \to \infty : \hat{Q}_t(a) \to Q(a)$$

Bayesian

- Bayesian bandits exploit prior knowledge of rewards, $p[\mathcal{R}]$
- They compute posterior distribution of rewards $p[\mathcal{R} \mid h_t]$

(Probability distribution to describe the uncertainty of parameters.)

Thompson Sampling

二、 Algorithm

Bernoulli Bandit

- K actions mean rewards $\theta = (\theta_1, \ldots, \theta_K)$
- one with probability θ_k $\mathbb{P}[r_1 = 1 | a_1, \theta] = \theta_{a_1}$
- zero with probability $1 - \theta_k$ $\mathbb{P}[r_1 = 0 | a_1, \theta] = 1 - \theta_{a_1}$
- Each θ_k: Take these priors to be beta-distributed with parameters

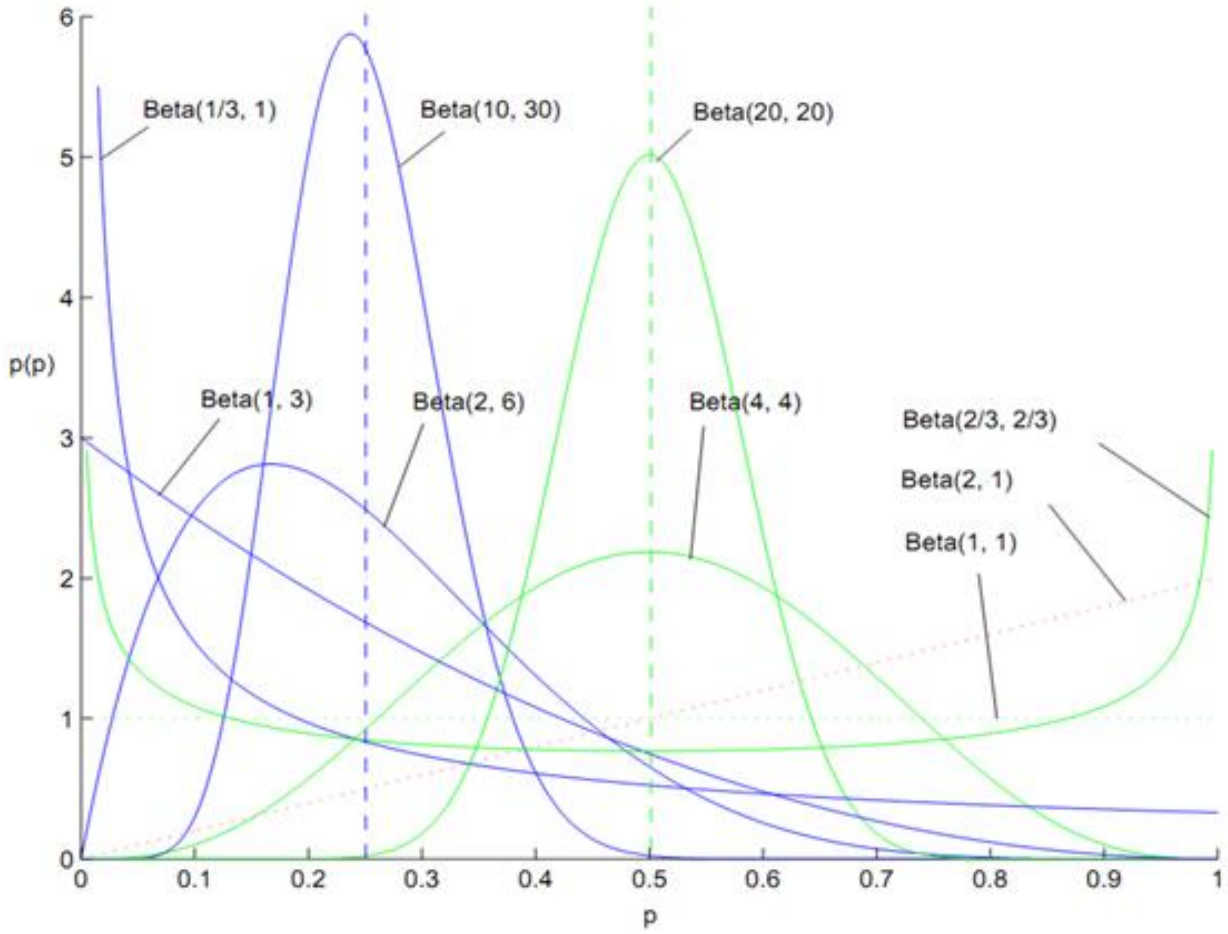
  $\alpha = (\alpha_1, \ldots, \alpha_K)$ and $\beta \in (\beta_1, \ldots, \beta_K)$.

二、 Algorithm

$\boldsymbol{BETA(\alpha, \beta)}$

$$p(\theta_k) = \frac{\Gamma(\alpha_k + \beta_k)}{\Gamma(\alpha_k)\Gamma(\beta_k)} \theta_k^{\alpha_k - 1}(1 - \theta_k)^{\beta_k - 1},$$

$$\Gamma(x) = \int_0^{\infty} t^{x-1} e^{-t} dt$$



二、Algorithm

DaSE Data Science & Engineering

共轭分布

$$p(\theta|reward) = \frac{p(reward|\theta)p(\theta)}{p(reward)} \propto p(reward|\theta)p(\theta) = Bernoulli(\theta)p(\theta)$$

$$p(reward|\theta) \sim Bernoulli(\theta)$$

$$p(\theta) \sim Beta(\alpha, \beta)$$

在贝叶斯统计中，Beta分布是Bernoulli分布的**共轭先验，**即在先验分布为Beta分布而似然函数为Bernoulli分布时，后验概率分布仍然是Beta分布。

$$\Rightarrow \quad Bernoulli(\theta)p(\theta) \sim Beta(\alpha, \beta)$$

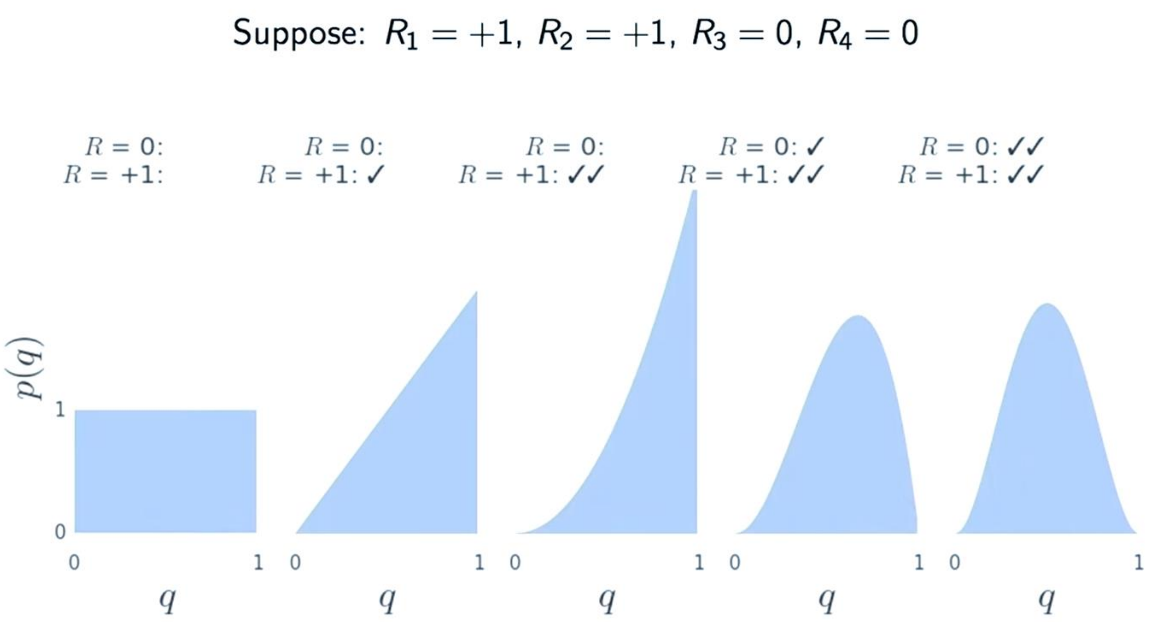
DaSE Data Science & Engineering

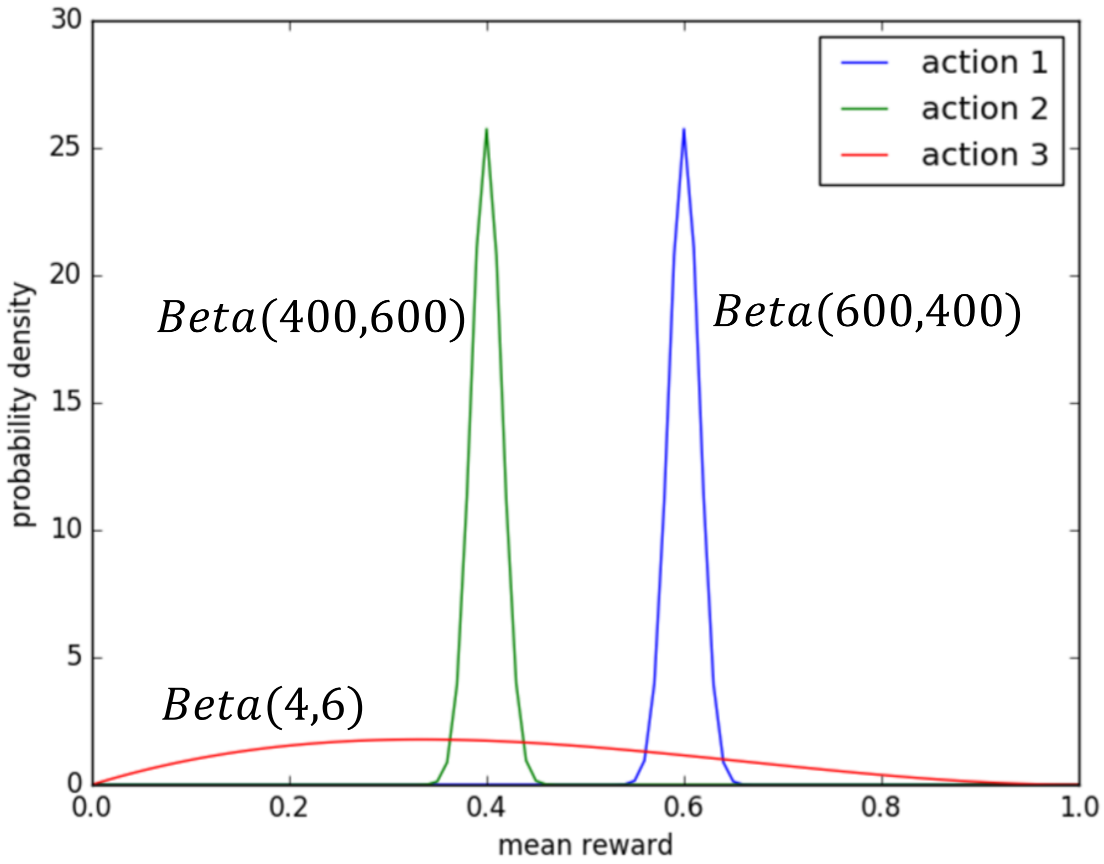
5. *Thompson Sampling*

- Consider bandits with Bernoulli reward distribution: rewards are 0 or $+1$
- For each action, the prior could be a uniform distribution on $[0, 1]$
- This means we think each mean reward in $[0, 1]$ is equally likely
- The posterior is a Beta distribution $\text{Beta}(x_a, y_a)$ with initial parameters $x_a = 1$ and $y_a = 1$ for each action a
- Updating the posterior:
  - $x_{A_t} \leftarrow x_{A_t} + 1$ when $R_t = 0$
  - $y_{A_t} \leftarrow y_{A_t} + 1$ when $R_t = 1$

二、 Algorithm

5. *Thompson Sampling*





二、 Algorithm

5. *Thompson Sampling*

Algorithm BernThompson(K, α, β)

for $t = 1, 2, \ldots$ **do**
    #sample model:
    for $k = 1, \ldots, K$ **do**
        Sample $\hat{\theta}_k \sim \text{beta}(\alpha_k, \beta_k)$
    end for

    #select and apply action:
    $x_t \leftarrow \text{argmax}_k \, \hat{\theta}_k$
    Apply x_t and observe r_t

    #update distribution:
    $(\alpha_{x_t}, \beta_{x_t}) \leftarrow (\alpha_{x_t}, \beta_{x_t}) + (r_t, 1 - r_t)$
end for

general →

Algorithm Thompson$(\mathcal{X}, p, q, r)$

for $t = 1, 2, \ldots$ **do**
    #sample model:
    Sample $\hat{\theta} \sim p$

    #select and apply action:
    $x_t \leftarrow \text{argmax}_{x \in \mathcal{X}} \, \mathbb{E}_{q_{\hat{\theta}}}[r(y_t) | x_t = x]$
    Apply x_t and observe y_t

    #update distribution:
    $p \leftarrow \mathbb{P}_{p,q}(\theta \in \cdot | x_t, y_t)$
end for

二、 Algorithm

5. *Thompson Sampling*

Algorithm 1 BernGreedy(K, α, β)

for $t = 1, 2, \ldots$ **do**
    #estimate model:
    for $k = 1, \ldots, K$ **do**
        $\hat{\theta}_k \leftarrow \alpha_k / (\alpha_k + \beta_k)$
    end for

    #select and apply action:
    $x_t \leftarrow \text{argmax}_k \hat{\theta}_k$
    Apply x_t and observe r_t

    #update distribution:
    $(\alpha_{x_t}, \beta_{x_t}) \leftarrow (\alpha_{x_t}, \beta_{x_t}) + (r_t, 1 - r_t)$
end for

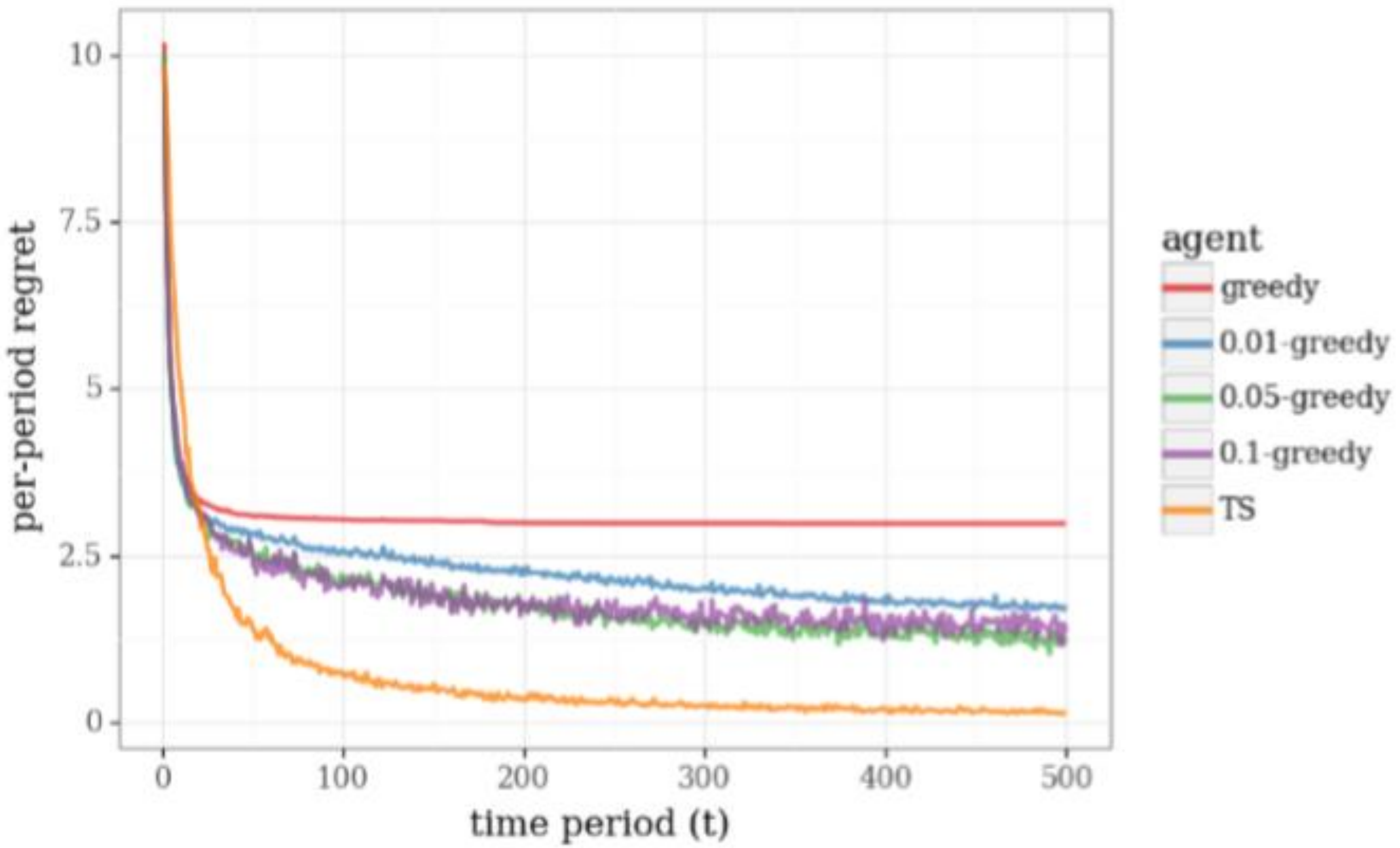
Algorithm 2 BernThompson(K, α, β)

for $t = 1, 2, \ldots$ **do**
    #sample model:
    for $k = 1, \ldots, K$ **do**
        Sample $\hat{\theta}_k \sim \text{beta}(\alpha_k, \beta_k)$
    end for

    #select and apply action:
    $x_t \leftarrow \text{argmax}_k \hat{\theta}_k$
    Apply x_t and observe r_t

    #update distribution:
    $(\alpha_{x_t}, \beta_{x_t}) \leftarrow (\alpha_{x_t}, \beta_{x_t}) + (r_t, 1 - r_t)$
end for

5. *Thompson Sampling*



A Tutorial on Thompson Sampling

DaSE Data Science & Engineering

6. *Gradient Bandit Algorithms*

- We consider learning a numerical preference for each action a, which we denote Ht(a).
- The larger the preference, the more often that action is taken, but the preference has no interpretation in terms of reward.
- action probabilities are determined according to a soft-max distribution:

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

- Initially all action preferences are the same (e.g., H1(a) = 0, for all a) so that all actions have an equal probability of being selected.

DaSE Data Science & Engineering

6. *Gradient Bandit Algorithms*

- On each step, after selecting action At and receiving the reward Rt, the action preferences are updated by:

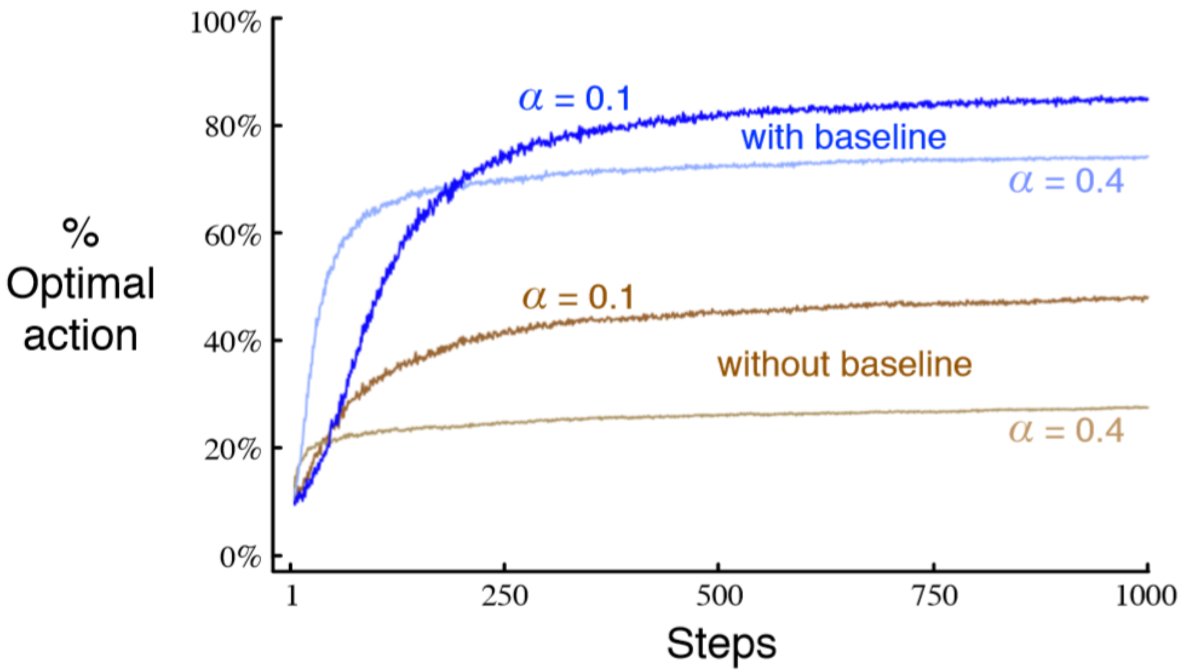
$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), \qquad \text{and}$$
$$H_{t+1}(a) \doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), \qquad \text{for all } a \neq A_t$$

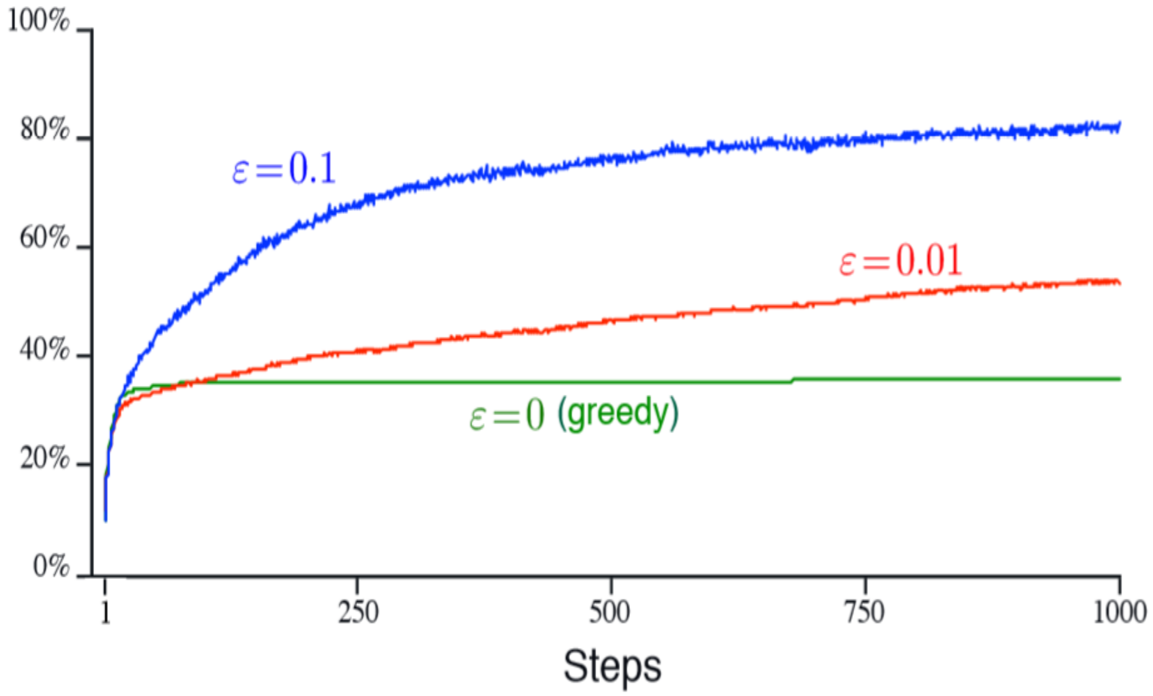
$\alpha > 0$ is a step-size parameter

$\bar{R}_t \in \mathbb{R}$ is the average of all the rewards (baseline)

DaSE Data Science & Engineering

6. *Gradient Bandit Algorithms*





Conclusion

- **Introduction**
- **Algorithm**
  - $\boldsymbol{Greedy}$
  - $\boldsymbol{\epsilon - Greedy}$
  - $\boldsymbol{Delay\ \epsilon - Greedy}$
  - $\boldsymbol{UCB}$
  - $\boldsymbol{Thompson\ Sampling}$
  - $\boldsymbol{Gradient\ Bandit\ Algorithms}$
- **Conclusion**

Reference

- **Introduction to Multi-Armed Bandits.**
- **A Tutorial on Thompson Sampling.**
- **Finite-time Analysis of the Multiarmed Bandit Problem.**
- **The book of Reinforcement Learning by Sutton.**
- **The course of Deep Reinforcement Learning by David Silver.**

THANKS