Lecture 4: Bandit Learning

Ziyu Shao

School of Information Science and Technology
ShanghaiTech University

March 28, 2025

Outline

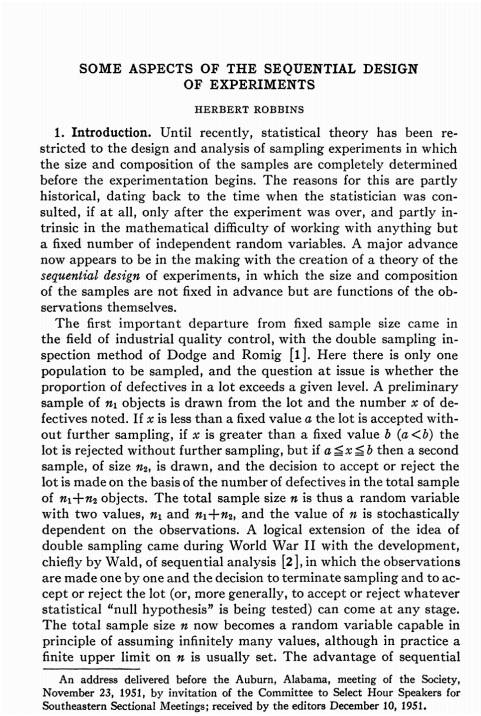
Outline

One-Armed Bandit

Multi-Armed Bandit

History

- 1933: William R. Thompson proposed the first algorithm (Thompson sampling method)
- 1952: Herbert Robbins (1915-2001) proposed the mathematical formulation of multi-armed bandit problem and sequential sampling algorithm for 2-armed bandit problem.



SOME ASPECTS OF THE SEQUENTIAL DESIGN OF EXPERIMENTS

HERBERT ROBBINS

1. **Introduction.** Until recently, statistical theory has been restricted to the design and analysis of sampling experiments in which the size and composition of the samples are completely determined before the experimentation begins. The reasons for this are partly historical, dating back to the time when the statistician was consulted, if at all, only after the experiment was over, and partly intrinsic in the mathematical difficulty of working with anything but a fixed number of independent random variables. A major advance now appears to be in the making with the creation of a theory of the *sequential design* of experiments, in which the size and composition of the samples are not fixed in advance but are functions of the observations themselves.

The first important departure from fixed sample size came in the field of industrial quality control, with the double sampling inspection method of Dodge and Romig [1]. Here there is only one population to be sampled, and the question at issue is whether the proportion of defectives in a lot exceeds a given level. A preliminary sample of n_1 objects is drawn from the lot and the number x of defectives noted. If x is less than a fixed value a the lot is accepted without further sampling, if x is greater than a fixed value b $(a<b)$ the lot is rejected without further sampling, but if $a \leqq x \leqq b$ then a second sample, of size n_2, is drawn, and the decision to accept or reject the lot is made on the basis of the number of defectives in the total sample of n_1+n_2 objects. The total sample size n is thus a random variable with two values, n_1 and n_1+n_2, and the value of n is stochastically dependent on the observations. A logical extension of the idea of double sampling came during World War II with the development, chiefly by Wald, of sequential analysis [2], in which the observations are made one by one and the decision to terminate sampling and to accept or reject the lot (or, more generally, to accept or reject whatever statistical "null hypothesis" is being tested) can come at any stage. The total sample size n now becomes a random variable capable in principle of assuming infinitely many values, although in practice a finite upper limit on n is usually set. The advantage of sequential

An address delivered before the Auburn, Alabama, meeting of the Society, November 23, 1951, by invitation of the Committee to Select Hour Speakers for Southeastern Sectional Meetings; received by the editors December 10, 1951.

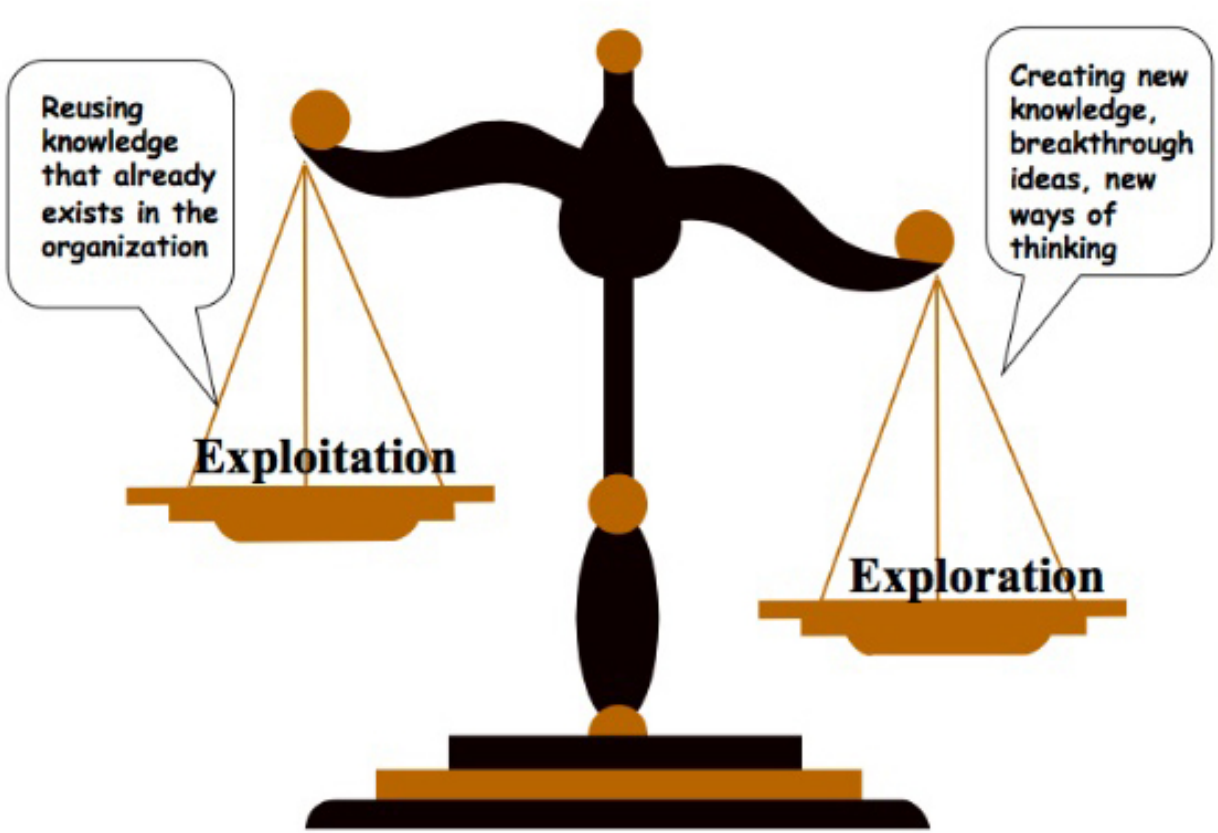
Why Bandit?

- A perfect model for online decision making under uncertainty
- Isolate an important component of reinforcement learning: exploration-vs-exploitation
- Rich and beautiful mathematically
- Widely applications

Exploration vs. Exploitation Dilemma

- Online decision-making involves a fundamental choice
  - **Exploitation**: staying with the option that gave the highest payoffs in the past
  - **Exploration**: exploring new options that might give higher payoffs in the future
- The best long-term strategy may involve short-term sacrifices
- Gather enough information to make the best overall decisions

Exploration vs. Exploitation Dilemma



Examples in Real Life

- Restaurant Selection
  - **Exploitation**: Go to your favorite restaurant
  - **Exploration**: Try a new restaurant
- Online Banner 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

Example of A Two-Armed Bandit

Time	1	2	3	4	5	6	7	8	9	10	11	12
Left arm	$1	$0			$1	$1	$0					
Right arm			$1	$0								

Left : $\frac{3}{5} = 0.6$ ✓ exploitation

Right : $\frac{1}{2}$ 0.5.

✓ : exploration

Widely Applications

- Several companies including Adobe, Amazon, Facebook, Google, LinkedIn, Microsoft, Netflix, and Twitter
- Clinical trials/dose discovery
- Recommendation systems (movies/news/etc)
- Advertisement placement
- A/B testing
- Monte Carlo tree search algorithm (game playing including AlphaGo)
- Resource allocation
- Network routing
- Dynamic pricing (e.g.,for Amazon products)
- Ranking (e.g.,for search)

Remark

Outline

Bandit: Special Case of Reinforcement Learning

- What is Reinforcement Learning?
- **Wikipedia**: reinforcement learning is an area of machine learning inspired by behavioral psychology, concerned with how **agents** ought to take **actions** in an **environment** so as to maximize some notion of cumulative **reward**.

Reward: Important Concept

- A **reward** R_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- The agent's job is to maximize cumulative reward

Reinforcement learning is based on the reward hypothesis:

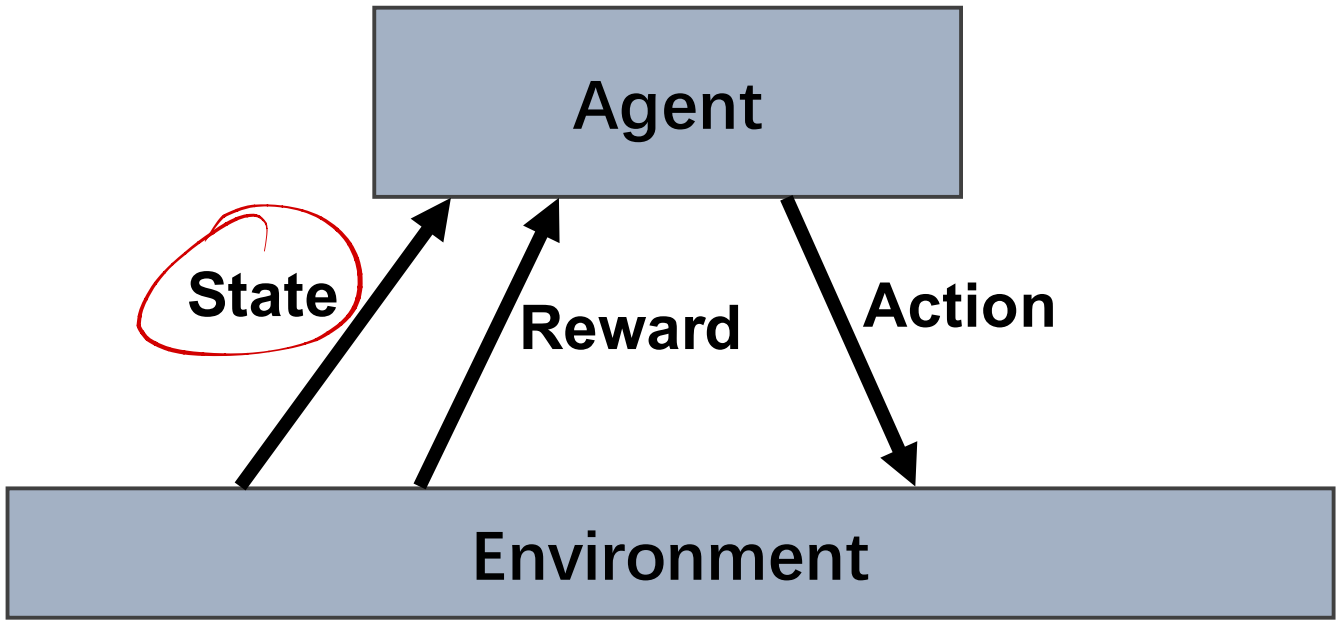
Definition

All goals can be described by the maximization of expected cumulative reward

Why Expectation of Rewards?

- Decision makers who base their decisions on expected values are called "risk-neutral"
- Von Neumann-Morgenstern expected utility theorem: with axioms of rational behavior under uncertainty
- A rational decision maker must choose amongst alternatives by computing the expected utility of the outcomes.

Reinforcement Learning Problems



Goal: Learn to choose actions that maximize rewards

Decision Making Under Uncertainty

	Actions don't change state of the world	Actions change state of the world
Learn model of outcomes	Multi-armed bandits	Reinforcement Learning
Given model of stochastic outcomes	Decision theory	Markov Decision Process

Three Running Examples of Bandits

- **News website.** When a new user arrives, a website site picks an article header to show, observes whether the user clicks on this header. The site's goal is maximize the total number of clicks.
- **Dynamic pricing.** A store is selling a digital good, e.g., an app or a song. When a new customer arrives, the store chooses a price offered to this customer. The customer buys (or not) and leaves forever. The store's goal is to maximize the total profit.
- **Investment.** Each morning, you choose one stock to invest into, and invest \$1. In the end of the day, you observe the change in value for each stock. The goal is to maximize the total wealth.

Examples: Action & Reward

Example	Action	Reward
News website	an article to display	1 if clicked, 0 otherwise
Dynamic pricing	a price to offer	p if sale, 0 otherwise
Investment	a stock to invest into	change in value during the day

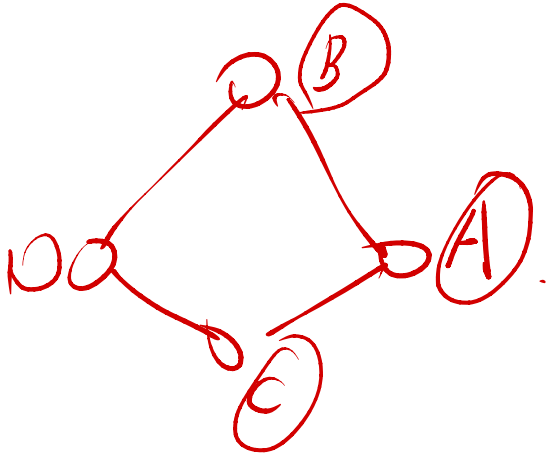
Problem Space

- Auxiliary Feedback
- Rewards Model
- Contexts
- Bayesian Priors
- Structured Rewards
- Global Constraints
- Structured Actions

Problem Space: Auxiliary Feedback

Example	Auxiliary feedback	Rewards for any other arms?
News website	N/A	no (*bandit feedback*).
Dynamic pricing	sale $\Rightarrow$ sale at any lower price, no sale $\Rightarrow$ no sale at any higher price	yes, for some arms, but not for all (*partial feedback*).
Investment	change in value for all other stocks	yes, for all arms (*full feedback*).

Three Types of Feedback



pick Arm A

observe reward of A

also reward of B, C

- **Bandit Feedback**: when the algorithm observes the reward for the chosen arm, and no other feedback
- **Full Feedback**: when the algorithm observes the rewards for all arms that could have been chosen
- **Partial Feedback**: when some information is revealed in addition to the reward of the chosen arm

Problem Space: Rewards Model

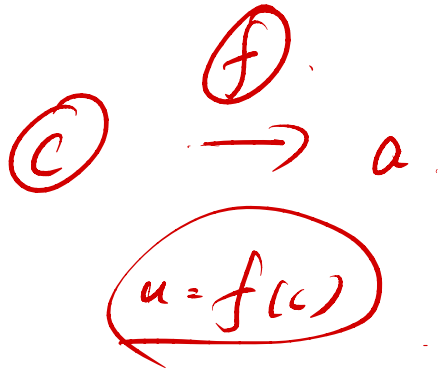
- **IID Rewards**: the reward for each arm is drawn independently from a fixed distribution that depends on the arm but not on the round t.
- **Stochastic Rewards (beyond IID)**: rewards evolves over time as a random process, e.g., a random walk.
- **Adversarial Rewards**: rewards can be arbitrary, as if they are chosen by an "adversary" that tries to fool the algorithm.
- **Constrained Adversary**: rewards are chosen by an adversary that is subject to some constraints, e.g., reward of each arm cannot change much from one round to another, or the reward of each arm can change at most a few times, or the total change in rewards is upper-bounded.

Problem Space: Contexts

- In each round, an algorithm may observe some context before choosing an action.
- Such context often comprises the known properties of the current user
- Allows for personalized actions

Example	Context
News website	user location and demographics
Dynamic pricing	customer's device (*e.g.*, cell or laptop), location, demographics
Investment	current state of the economy.

Problem Space: Contexts



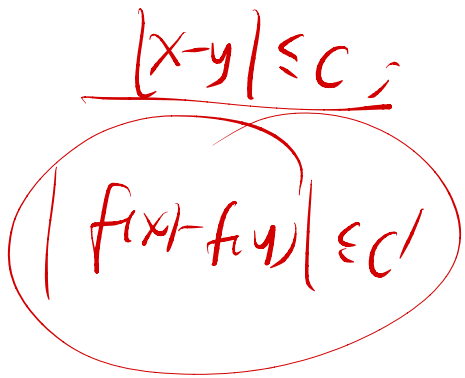
- The algorithm has a different high-level objective.
- It is no longer interested in learning one good arm, since any one arm may be great for some contexts, and terrible for some others.
- Instead, it strives to learn the best policy which maps contexts to arms
- While not spending too much time exploring.

Problem Space: Bayesian Priors

- Each problem can be studied under a Bayesian approach, whereby the problem instance comes from a known distribution (called Bayesian prior).
- One is typically interested in provable guarantees in expectation over this distribution.

Problem Space: Structured Rewards

- Rewards may have a known structure, e.g., arms correspond to points in $\mathbb{R}^d$,
- And in each round the reward is a linear (resp., concave or Lipschitz) function of the chosen arm.



Problem Space: Global Constraints

- The algorithm can be subject to global constraints that bind across arms and across rounds.
- For example, in dynamic pricing there may be a limited inventory of items for sale.

Problem Space: Structured Actions

- An algorithm may need to make several decisions at once,
- e.g., a news website may need to pick a slate of articles,
- and a seller may need to choose prices for the entire slate of offerings.

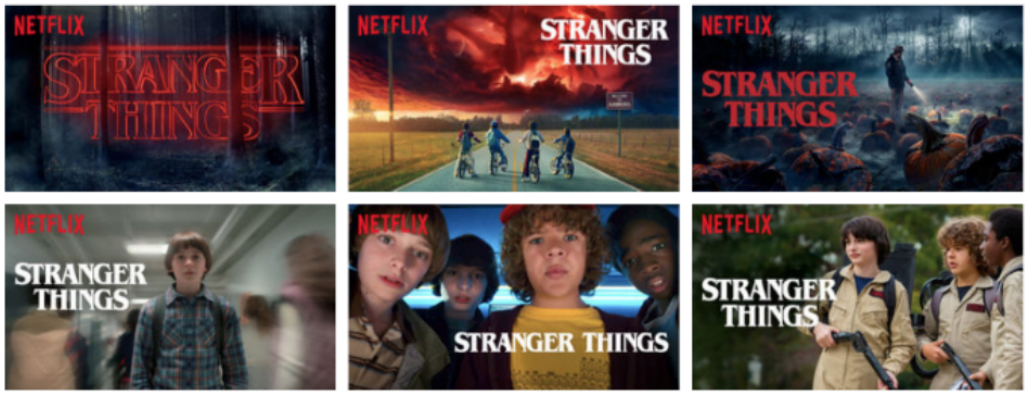
Application Domains

Application domain	Action	Reward
medical trials	which drug to prescribe	health outcome.
web design	*e.g.*, font color or page layout	#clicks.
content optimization	which items/articles to emphasize	#clicks.
web search	search results for a given query	1 if the user is satisfied.
advertisement	which ad to display	revenue from ads.
recommender systems	*e.g.*, which movie to watch	1 if follows recommendation.
sales optimization	which products to offer at which prices	revenue.
procurement	which items to buy at which prices	#items procured
auction/market design	*e.g.*, which reserve price to use	revenue
crowdsourcing	which tasks to give to which workers, and at which prices	1 if task completed at sufficient quality.
datacenter design	*e.g.*, which server to route the job to	job completion time.
Internet	*e.g.*, which TCP settings to use?	connection quality.
radio networks	which radio frequency to use?	1 if successful transmission.
robot control	a "strategy" for a given task	job completion time.

Real-World Example: Netflix

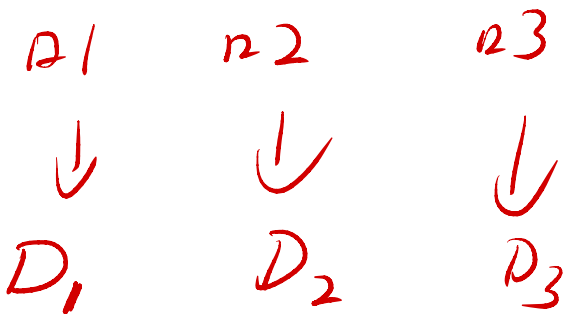
For a particular movie, we want to decide what image to show (to all the NEFLIX users)

- **Actions**: uploading one of the K movie posters to a user's home screen
- **Unknown true mean rewards**: the % of NETFLIX users that will click on the title and watch the movie
- **Estimated mean rewards**: the average click rate observed



Outline

Stochastic Bandits



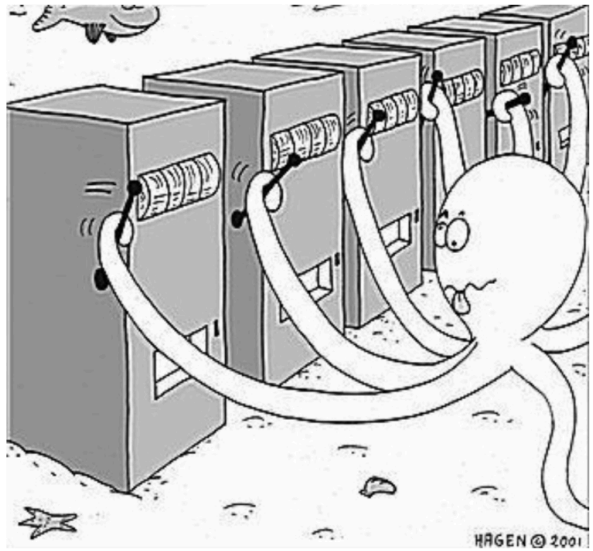
- Bandit feedback: observes only the reward for the selected action, and nothing else.
- IID reward for each action: every time one action a is chosen, the reward is sampled independently from the reward distribution associated with the action a.
- Per-round rewards are bounded.

Stochastic Bandits

step / round / stage

$E\left(\sum_{t=1}^{\infty} b^t r_t\right)$ $0<b<1$

- 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 maximize expectation of cumulative reward within t rounds $E(\sum_{\tau=1}^{t} r_\tau)$



T $E\left(\sum_{t=1}^{T} r_t\right)$

const.

Regret: Equivalent Goal

- The value of an arbitrary action a (action-value) is the mean reward for action a (unknown):

$$q_*(a) = Q(a) = E[r_t | a_t = a], \forall a \in \{1, 2, \ldots, K\}$$

- The *optimal value* V^* is

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

- The *regret* is the opportunity loss for one round τ

$$L_\tau = \mathbb{E}[V^* - Q(a_\tau)]$$

where

$$Q(a_\tau) = E[r_\tau | a_\tau].$$

Regret: Equivalent Goal

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

$$= t \cdot V^* - \sum_{\tau=1}^{t} \mathbb{E}[\mathbb{E}[r_\tau | a_\tau]] \overset{\text{Adam}}{=\!=} tV^* - \sum_{\tau=1}^{t} \mathbb{E}[r_\tau]$$

- The *total regret* is the total opportunity loss after the end of round t

$$= tV^* - \mathbb{E}\left[\sum_{\tau=1}^{t} r_\tau\right]$$

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

minimize L_t $\Leftrightarrow$ maximize

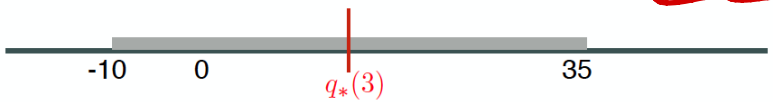
- Maximize the Expectation of Cumulative Reward $\equiv \mathbb{E}\left[\sum_{\tau=1}^{t} r_\tau\right]$
  Minimize Total Regret

Example on Action-Value

Arm

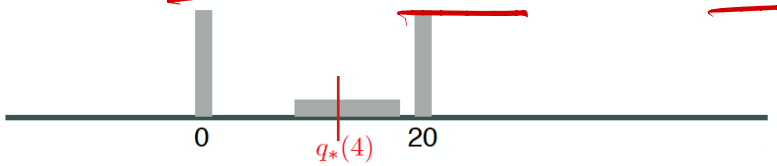
Reward Distribution

- Action 1 — Reward is always 8
  - value of action 1 is $q_*(1) = 8$
- Action 2 — 88% chance of 0, 12% chance of 100!
  - value of action 2 is $q_*(2) = .88 \times 0 + .12 \times 100 = 12$
- Action 3 — Randomly between -10 and 35, equiprobable



$q_*(3) = \frac{45}{2} = 22.5$

- Action 4 — a third 0, a third 20, and a third from {8,9,..., 18}



$q_*(4) = \frac{1}{3} \cdot 0 + \frac{1}{3} \cdot 20 + \frac{1}{3} \cdot \left(\frac{1}{11} \times 8 + \frac{1}{11} \times 9 + \cdots + \frac{1}{11} \times 18 \right)$

$= 11$

Example on Action-Value

- In general, those values are unknown.
- We must both try actions to learn the action-values (explore), and prefer those that appear best (exploit).

The Exploration/Exploitation Dilemma

Action a

1° $Q(a) = q_*(a)$ unknown

$\hat{Q}_t(a) \approx Q(a)$

estimation / learn.

2°. Define the greedy action at time t as:

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

if: $a_t = a_t^*$; making exploitation.

$a_t \neq a_t^*$; making exploration.

Action-Value Methods

Sample mean

$$\hat{Q}_t(a) = \frac{\sum_{\tau=1}^{t} r_\tau \cdot \mathbb{1}_{\{a_\tau = a\}}}{\sum_{\tau=1}^{t} \mathbb{1}_{\{a_\tau = a\}}} \xrightarrow{\text{SLLN}} Q(a).$$

- Methods that learn action-value estimates and nothing else
- For example, estimate action values as sample average
- The sample-average estimates converge to the true values if the action is taken an infinite number of times

Initialization stage: pull each arm once

Counting Regret

- The *count* $N_t(a)$ is the number of selections for action a after the end of round t: $N_t(a) = \sum_{\tau=1}^{t} \mathbb{I}_{a_\tau = a}$
- The gap Δ_a is the difference in value between action a and optimal action a^*, $\Delta_a = V^* - Q(a)$
- Regret is a function of gaps and the counts

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

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

Regret Decomposition Lemma

Lemma

$$L_t = \sum_{a \in \mathcal{A}} \mathbb{E}[N_t(a)] \Delta_a$$

time dimension $\Longleftrightarrow$ Action dimension.

Proof

1^0. $Q(a_t) = E[r_t | a_t]$ r.v. $\Rightarrow E[Q(a_t)] = E[E[r_t|a_t]] = E[r_t]$.

2^0. $S_t \triangleq \sum_{\tau=1}^{t} Q(a_\tau) \Rightarrow E(S_t) = \sum_{\tau=1}^{t} E[Q(a_\tau)] = \sum_{\tau=1}^{t} E[r_\tau] = E(\sum_{\tau=1}^{t} r_\tau]$

since $\sum_{a \in A} \mathbb{1}_{\{a_\tau = a\}} = 1$ for any fixed τ.

$\Rightarrow E(S_t) = E(\sum_{\tau=1}^{t} r_\tau) = E(\sum_{\tau=1}^{t} r_\tau \cdot \underbrace{\sum_{a \in A} \mathbb{1}_{\{a_\tau = a\}}}_{1})$

$= E(\sum_{a \in A} \sum_{\tau=1}^{t} r_\tau \mathbb{1}_{\{a_\tau = a\}}) = \sum_{a \in A} \sum_{\tau=1}^{t} E[r_\tau \mathbb{1}_{\{a_\tau = a\}}]$ (*1)

3^0. On the other hand, $\sum_{\tau=1}^{t} (\sum_{a \in A} \mathbb{1}_{\{a_\tau = a\}}) = \sum_{\tau=1}^{t} \cdot 1 = t$

$\Rightarrow E[\sum_{\tau=1}^{t} \sum_{a \in A} \mathbb{1}_{\{a_\tau = a\}}] = t \Rightarrow \sum_{\tau=1}^{t} \sum_{a \in A} E[\mathbb{1}_{\{a_\tau = a\}}] = t$

$\Rightarrow \sum_{a \in A} \sum_{\tau=1}^{t} E[\mathbb{1}_{\{a_\tau = a\}}] = t$ (*2)

Proof

4°. Now the total regret

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

$$= tv^* - E(S_t) \overset{(*1)}{=} tv^* - \sum_{a\in A}\sum_{\tau=1}^{t} E[r_\tau \mathbb{1}_{\{a_\tau=a\}}]$$

$$\overset{(*2)}{=} \sum_{a\in A}\sum_{\tau=1}^{t} E[\mathbb{1}_{\{a_\tau=a\}}]v^* - \sum_{a\in A}\sum_{\tau=1}^{t} E[r_\tau \mathbb{1}_{\{a_\tau=a\}}]$$

$$= \sum_{a\in A}\sum_{\tau=1}^{t} E[(v^* - r_\tau)\mathbb{1}_{\{a_\tau=a\}}] \quad (*3)$$

5°. $E[(v^* - r_\tau)\mathbb{1}_{\{a_\tau=a\}} \mid a_\tau] = \mathbb{1}_{\{a_\tau=a\}} E[(v^* - r_\tau)\mid a_\tau]$

$$= \mathbb{1}_{\{a_\tau=a\}}\cdot(v^* - E[r_\tau \mid a_\tau]) = \mathbb{1}_{\{a_\tau=a\}}(v^* - Q(a_\tau))$$

$$= \mathbb{1}_{\{a_\tau=a\}}(v^* - Q(a)) = \mathbb{1}_{\{a_\tau=a\}}\cdot\Delta_a$$

$\Rightarrow E[(v^* - r_\tau)\mathbb{1}_{\{a_\tau=a\}}] = E[\quad] = E[\mathbb{1}_{\{a_\tau=a\}}\cdot\Delta_a] \quad (*4)$

Proof

6°. Then we have

$$
\begin{aligned}
Lt &\overset{(*3)}{=} \sum_{a \in A} \sum_{\tau=1}^{t} E\left[(v^* - r_\tau) \mathbb{1}_{\{a_\tau = a\}} \right] \\
&\overset{(*4)}{=} \sum_{a \in A} \sum_{\tau=1}^{t} E\left[\mathbb{1}_{\{a_\tau = a\}} \cdot \Delta_a \right] \\
&= \sum_{a \in A} E\left[\sum_{\tau=1}^{t} \mathbb{1}_{\{a_\tau = a\}} \cdot \Delta_a \right] \\
&= \sum_{a \in A} E\left[\left(\sum_{\tau=1}^{t} \mathbb{1}_{\{a_\tau = a\}} \right) \right] \cdot \Delta_a \\
&= \sum_{a \in A} E[N_t(a)] \cdot \Delta_a
\end{aligned}
$$

$$
N_t(a) \triangleq \sum_{\tau=1}^{t} \mathbb{1}_{\{a_\tau = a\}}
$$

Landau Notation

- Given functions $f, g : \mathbb{N} \to [0, \infty)$, we define

$$f(n) = O(g(n)) \Leftrightarrow \limsup_{n\to\infty} \frac{f(n)}{g(n)} < \infty\,.$$

$$f(n) = o(g(n)) \Leftrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0\,.$$

$$f(n) = \Omega(g(n)) \Leftrightarrow \liminf_{n\to\infty} \frac{f(n)}{g(n)} > 0\,.$$

$$f(n) = \omega(g(n)) \Leftrightarrow \liminf_{n\to\infty} \frac{f(n)}{g(n)} = \infty\,.$$

$$f(n) = \Theta(g(n)) \Leftrightarrow f(n) = O(g(n)) \text{ and } f(n) = \Omega(g(n))\,.$$

Good Learners

- Let L_n denote the regret over n rounds
- A good learner achieves sublinear regret: $L_n = o(n)$ or equivalently $\lim_{n\to\infty} L_n/n = 0$

Linear or Sublinear Regret

- If an algorithm forever explores it will have linear total regret
- If an algorithm never explores it will have linear total regret
- Is it possible to achieve sublinear total regret?

Outline

Greedy Algorithm

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

$$\hat{Q}_t(a) = \frac{1}{N_{t-1}(a)} \sum_{\tau=1}^{t-1} r_\tau \mathbf{1}(a_\tau = a)$$

Avoid $N_t(a) = 0$

- The *greedy* algorithm selects action with highest value

$$a_t^* = \arg\max_{a \in \mathcal{A}} \hat{Q}_t(a)$$

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

ϵ-Greedy Algorithm

- The ϵ-*greedy* algorithm continues to explore forever
  - With probability $1 - \epsilon$ select $a_t^* = \arg\max_{a \in \mathcal{A}} \hat{Q}_t(a)$
  - With probability ϵ select a random action
- The we have the lower bound of the total regret

$$L_t \geq \frac{t \cdot \epsilon}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \Delta_a$$

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

Proof of the Lower Bound

1°. For any fixed t, $P(a_t = a_t^*) = 1-\varepsilon + \frac{\varepsilon}{|A|}$.

$$P(a_t = a) = \frac{\varepsilon}{|A|}, \quad \forall a \neq a_t^*.$$

$$\Rightarrow \forall a \in A, \quad P(a_t = a) \geq \frac{\varepsilon}{|A|}.$$

2°. $N_t(a) = \sum_{\tau=1}^{t} \mathbb{1}\{a_\tau = a\} \Rightarrow \mathbb{E}(N_t(a)) = \mathbb{E}\left(\sum_{\tau=1}^{t} \mathbb{1}\{a_\tau = a\}\right)$

$$= \sum_{\tau=1}^{t} \mathbb{E}[\mathbb{1}\{a_\tau = a\}] = \sum_{\tau=1}^{t} P(a_\tau = a) \geq \sum_{\tau=1}^{t} \frac{\varepsilon}{|A|} = \frac{t\cdot\varepsilon}{|A|}.$$

3°. By regret decomposition lemma,

$$L_t = \sum_{a \in A} \mathbb{E}[N_t(a)] \cdot \Delta_a \geq \sum_{a \in A} \frac{t\cdot\varepsilon}{|A|} \Delta_a = \frac{t\cdot\varepsilon}{|A|} \cdot \sum_{a \in A} \Delta_a$$

✓

Incremental Implementation

1°. $\hat{Q}_n = \frac{1}{n-1}(R_1 + \cdots + R_{n-1})$

2°. Incremental implementation

$$\Rightarrow \hat{Q}_{n+1} = \hat{Q}_n + \frac{1}{n}(R_n - \hat{Q}_n)$$

feedback

$\hat{Q}_{n+1}$: new estimation

$\hat{Q}_n$: old estimation

$\frac{1}{n}$: Step-size

$R_n - \hat{Q}_n$: estimation error

Stochastic Approximation $\hat{Q}_{n+1} = \hat{Q}_n + \alpha(n)[R_n - \hat{Q}_n]$

Under mild condition:

$\alpha(n) \in \left\{\frac{1}{n}, \frac{c}{an+b}\right\}$

$$\left.\begin{array}{l} \sum_{n=1}^{\infty} \alpha_n = \infty \\ \sum_{n=1}^{\infty} \alpha_n^2 < \infty \end{array}\right\} \Rightarrow \text{Converges w.p. 1.}$$

Derivation of Incremental Update

1°. $\hat{Q}_{n+1} = \frac{1}{n}\sum_{i=1}^{n} R_i = \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right)$

2°. $\hat{Q}_n = \frac{1}{n-1}\sum_{i=1}^{n-1} R_i \Rightarrow \sum_{i=1}^{n-1} R_i = (n-1)\hat{Q}_n$

3°. $\hat{Q}_{n+1} = \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right)$

$= \frac{1}{n}\left(R_n + (n-1)\hat{Q}_n\right)$

$= \frac{1}{n}\left[(R_n - \hat{Q}_n) + n\hat{Q}_n\right]$

$= \frac{1}{n}\left[R_n - \hat{Q}_n\right] + \hat{Q}_n$

$= \hat{Q}_n + \frac{1}{n}\left[R_n - \hat{Q}_n\right]$

ϵ-Greedy Bandit Algorithm

A simple bandit algorithm

Initialize, for $a = 1$ to k:

$Q(a) \leftarrow 0$

$N(a) \leftarrow 0$

pull all arms once.

Repeat forever:

$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

$R \leftarrow bandit(A)$

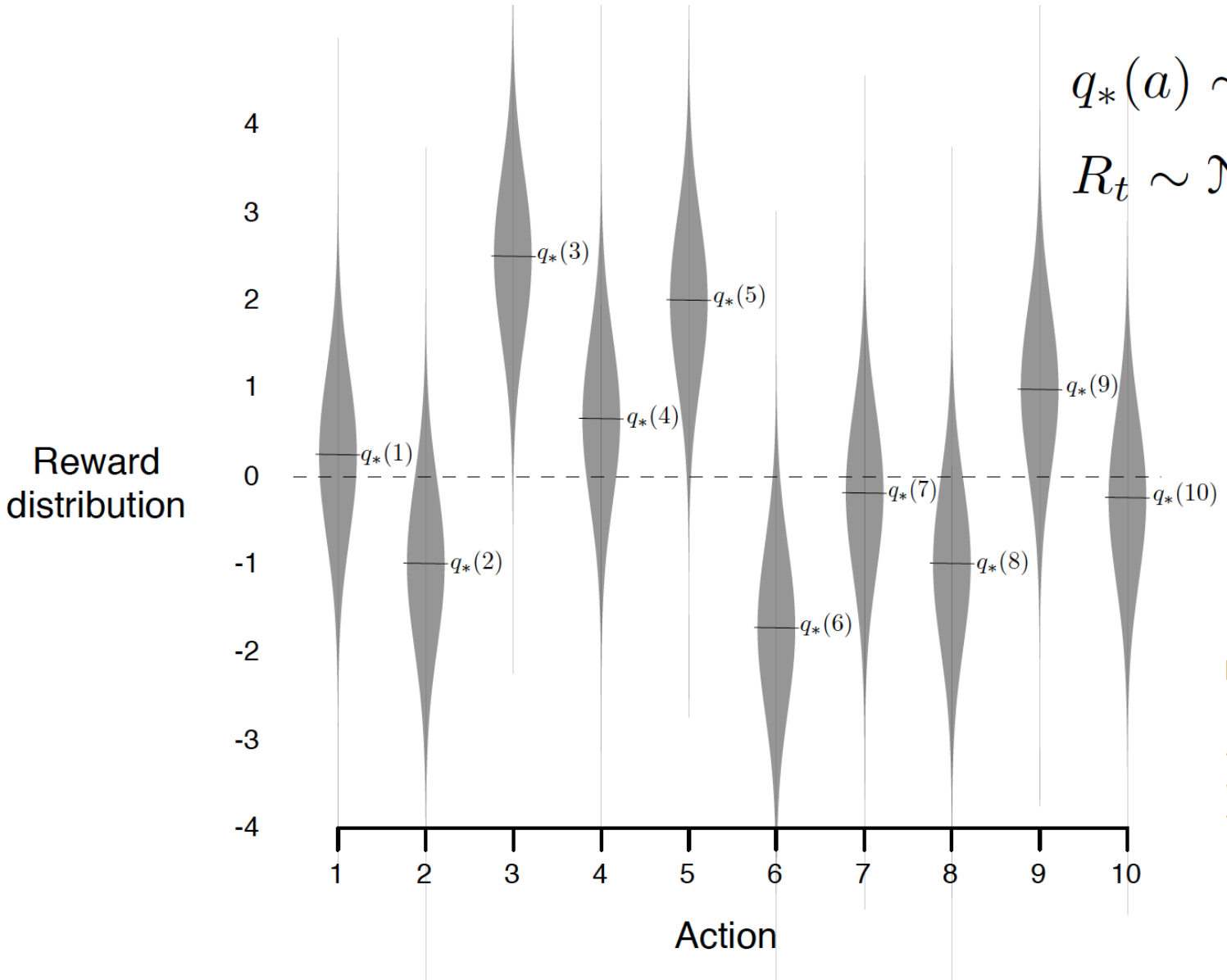
$N(A) \leftarrow N(A) + 1$

$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\left[R - Q(A)\right]$

Example: 10-armed Testbed

$$q_*(a) \sim \mathcal{N}(0, 1)$$

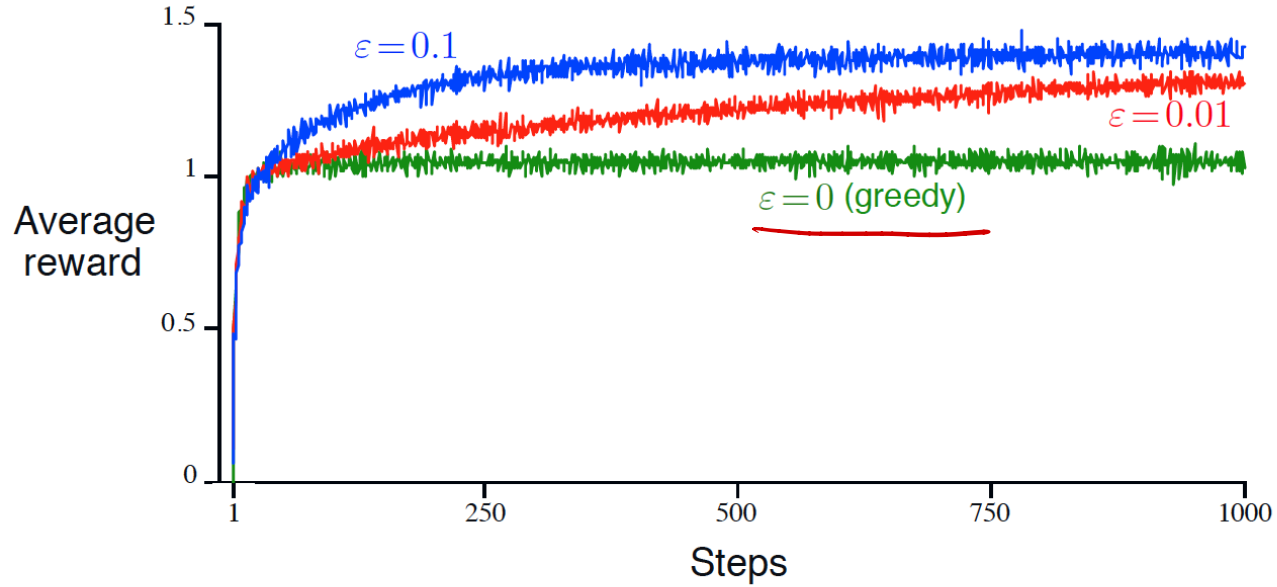
$$R_t \sim \mathcal{N}(q_*(a), 1)$$

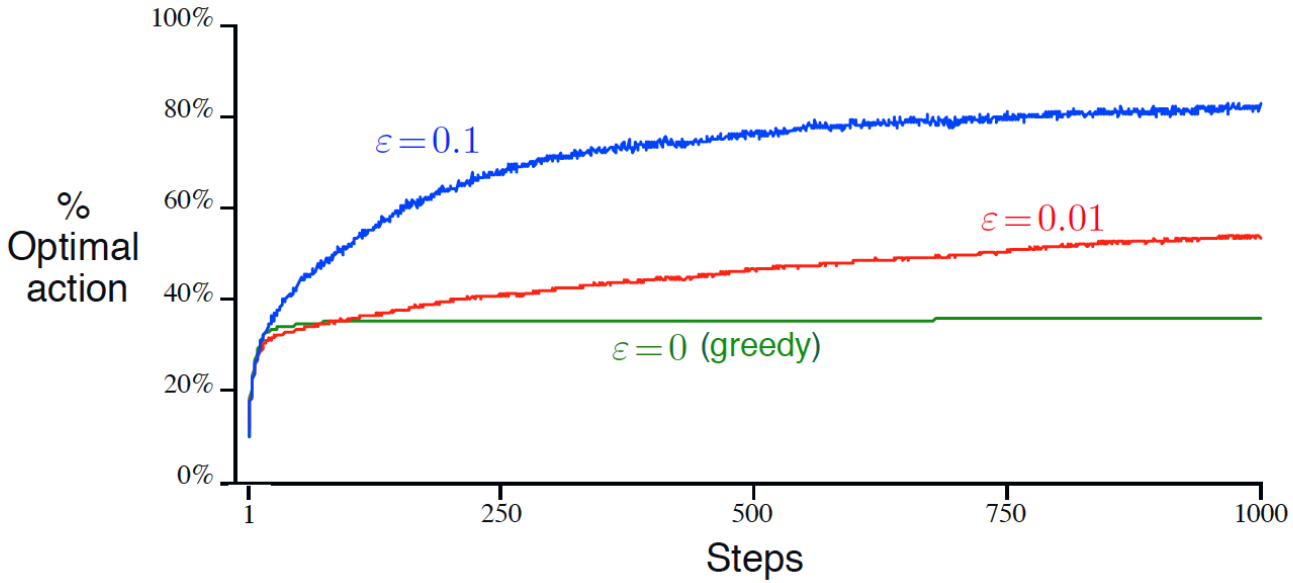


Run for 1000 steps

Repeat the whole thing 2000 times with different bandit tasks

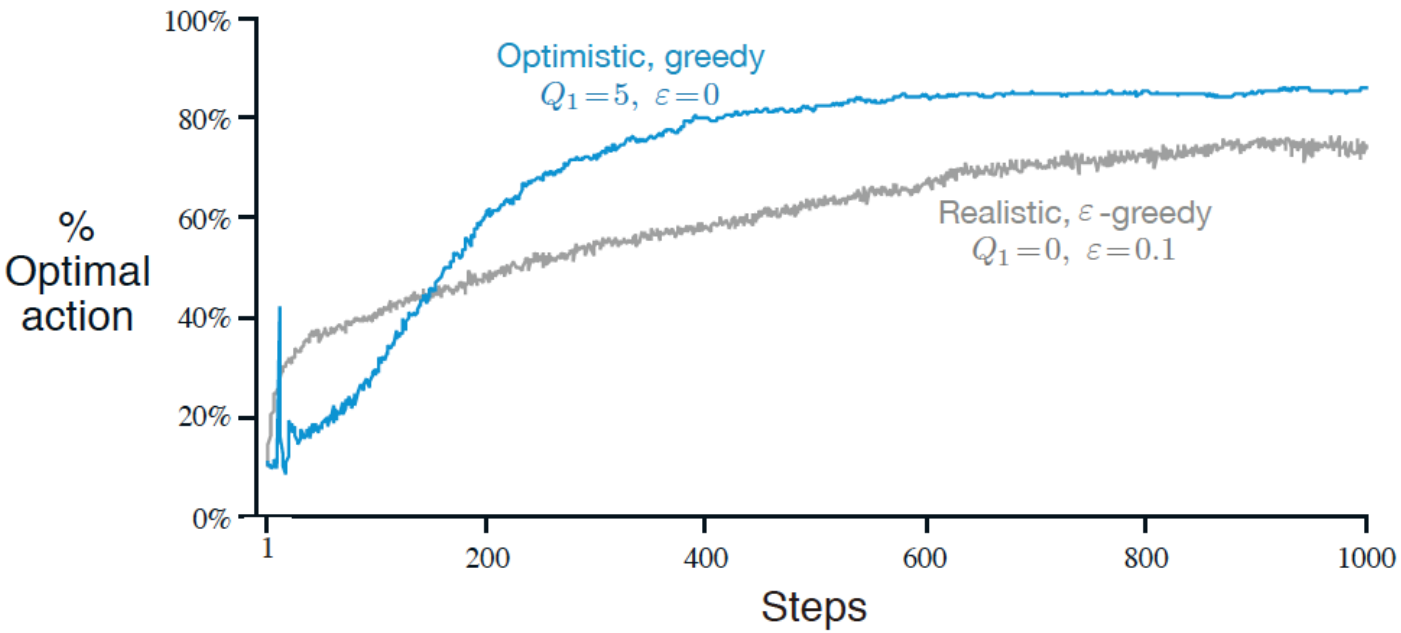
Performance of ϵ-Greedy





Optimistic Initial Values

- All methods so far depend on $\hat{Q}_1(a)$, i.e., the initial action-value estimate. So far we have used $\hat{Q}_1(a) = 0$.
- Suppose we initialize the action values optimistically ($\hat{Q}_1(a) = 5$), e.g., on the 10-armed testbed (with $\alpha = 0.1$)



Optimistic Initialization

- Simple and practical idea: initialize $\hat{Q}(a)$ to high values
- Update action value by incremental Monte-Carlo evaluation
- Starting with $N(a) > 0$

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

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

Decaying ϵ_t-Greedy Algorithm

- Pick a decay schedule for ϵ_1, ϵ_2, ...

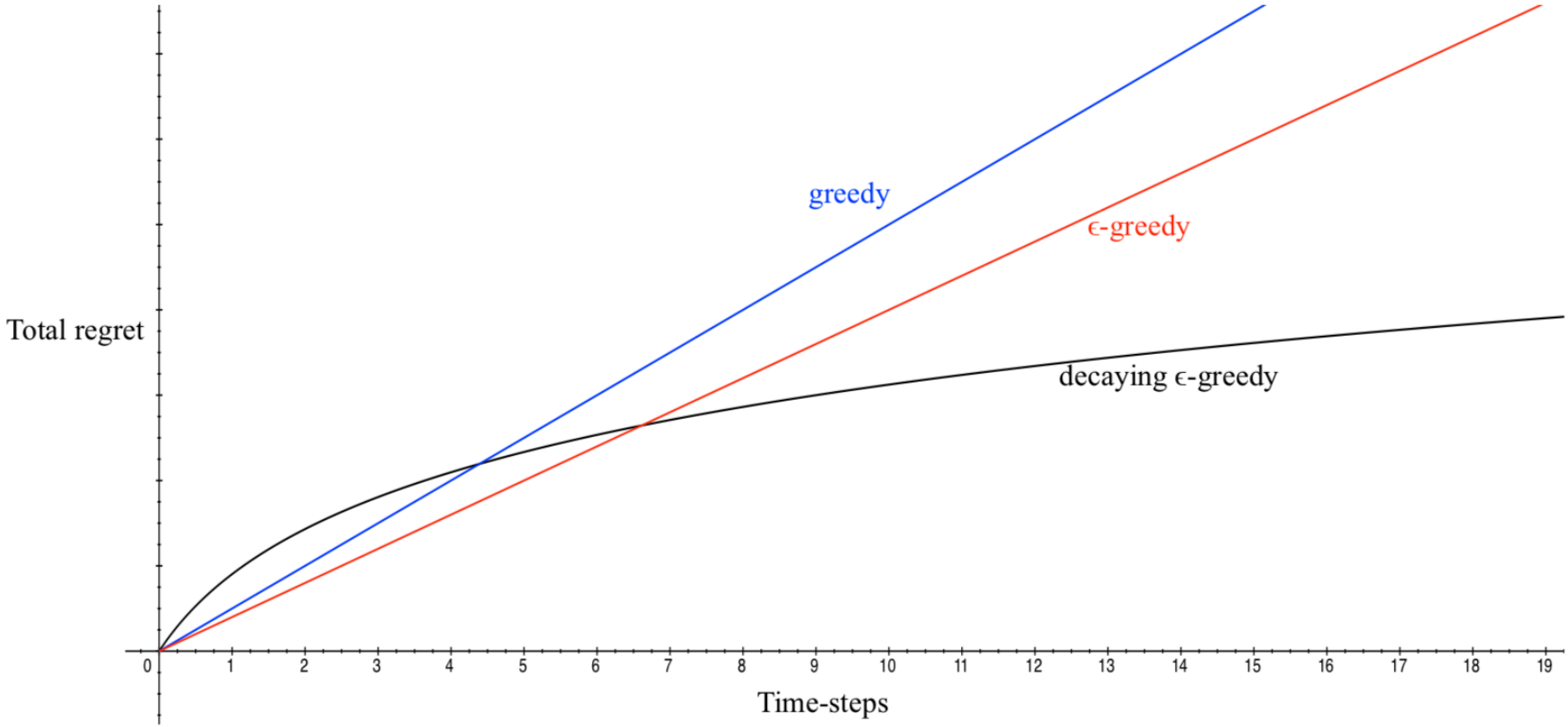
$$c > 0$$

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

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

- Consider the following schedule
- 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 $\mathcal{R}$)

Performance Comparison



Lower Bound

- The performance of any algorithm is determined by similarity between optimal arm and other arms
- This is described formally by the gap Δ_a and the similarity in distributions $KL(\mathcal{R}^a \| \mathcal{R}^{a^*})$

Theorem (Lai and Robbins)

Asymptotic total regret is at least logarithmic in number of steps

$$\lim_{t \to \infty} L_t \geq \log t \sum_{a | \Delta_a > 0} \frac{\Delta_a}{KL(\mathcal{R}^a \| \mathcal{R}^{a^*})}$$

Revisit Kullback-Leibler Divergence

- A finite sample space Ω
- Two probability distributions on Ω: $\mathbf{p}$ & $\mathbf{q}$
- Kullback-Leibler Divergence (KL-divergence) is defined as

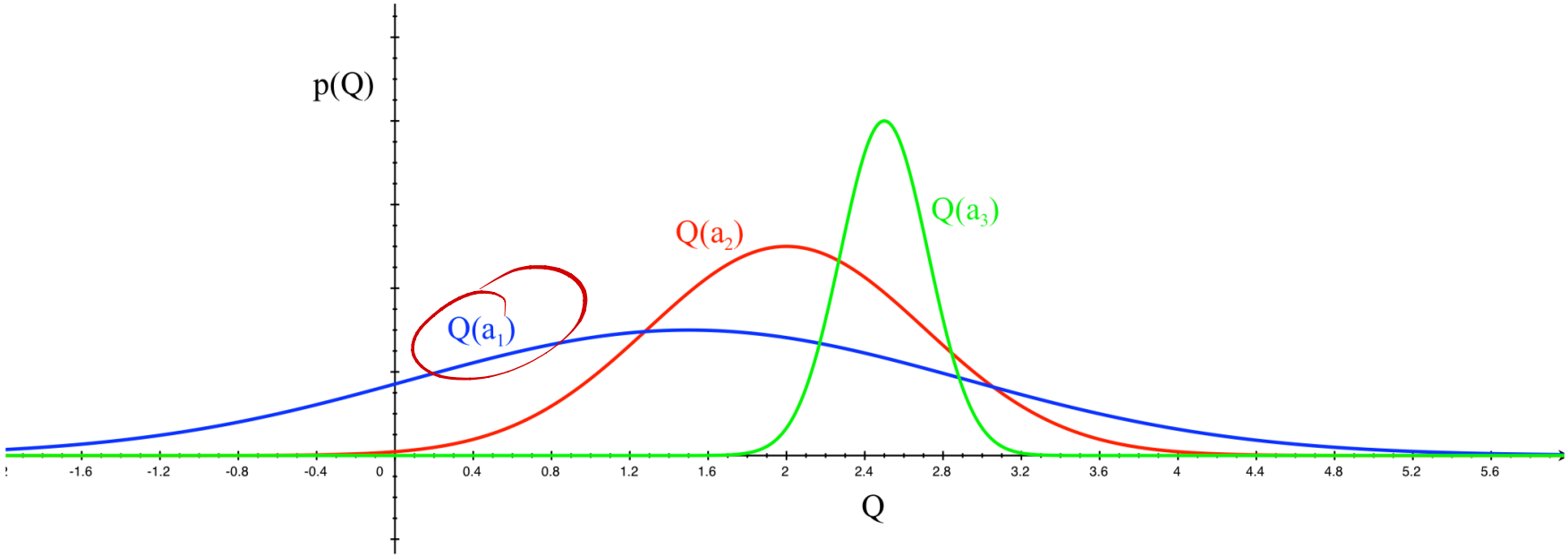
$$KL(\mathbf{p}, \mathbf{q}) = \sum_{x \in \Omega} p(x) \ln \frac{p(x)}{q(x)} = \mathbb{E}_{X \sim \mathbf{p}} \left[\ln \frac{p(X)}{q(X)} \right]$$

- Gibbs' Inequality: $KL(\mathbf{p}, \mathbf{q}) \geq 0$ for any two distributions $\mathbf{p}, \mathbf{q}$, with equality iff $\mathbf{p} = \mathbf{q}$.
- Pinsker's inequality: for any event $A \subset \Omega$, we have

$$2 \left(\mathbf{p}(A) - \mathbf{q}(A) \right)^2 \leq KL(\mathbf{p}, \mathbf{q}).$$

Outline

Optimism in the Face of Uncertainty



- Which action should we pick?
- The more uncertain we are about an action-value
- The more important it is to explore that action
- It could turn out to be the best action
- One should act as if the environment is as nice as plausibly possible

Upper Confidence Bounds

- Estimate an upper confidence bound $\hat{U}_t(a)$ for each action value
- Such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with high probability
- This depends on the number of times $N_t(a)$ 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 maximizing Upper Confidence Bound (UCB)

$$a_t = \arg\max_{a \in \mathcal{A}} \{\hat{Q}_t(a) + \hat{U}_t(a)\}.$$

exploitation

exploration

Hoeffding's Inequality

Theorem (Hoeffding's Inequality)

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

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

- We will apply Hoeffding's Inequality to rewards of the bandit
- Conditioned on selecting action a

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

$$\mathbb{P}\left[Q(a) \leq \hat{Q}_t(a) + U_t(a)\right] \geq 1 - e^{-2N_t(a)U_t^2(a)}$$

Calculating Upper Confidence Bounds

- Pick a probability p that true value exceeds UCB
- Now solve for $U_t(a)$

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

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

- Reduce p as we observe more rewards, e.g. $p = t^{-4}$
- Ensures we select optimal action as $t \to \infty$

$$P(Q_a \le \hat{Q}_t(a) + U_t(a)) \ge 1 - t^{-4} = 1 - \frac{1}{t^4} \to 1 \quad (t \to \infty)$$

$$U_t(a) = \sqrt{\frac{2\log t}{N_t(a)}}$$

UCB1

- This leads to the UCB1 algorithm: try each action (arm) once, then select actions according to the following rule

$$a_t = \arg\max_{a \in \mathcal{A}} \left\{ \hat{Q}_t(a) + c \sqrt{\frac{2 \log t}{N_t(a)}} \right\}$$

$t \gg 1$

$N_t(a) \ll 1$

- An action a is chosen at time t for two possible reasons
- Exploitation: $\hat{Q}_t(a)$ is large, indicating a high reward w.h.p
- Exploration: $N_t(a)$ is small, indicating that this action has not been explored much.
- Summing two parts up is a nature way to trade off them.

Performance of UCB1

Theorem

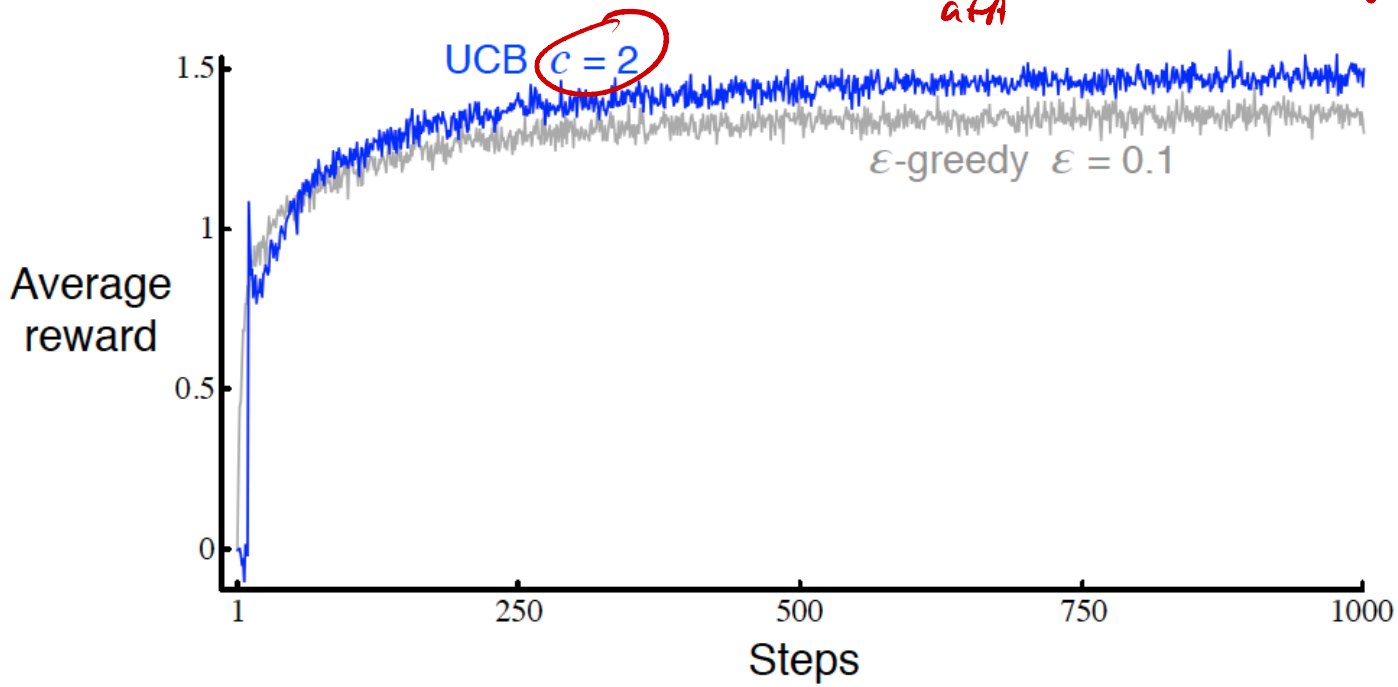
The UCB1 algorithm achieves logarithmic asymptotic total regret

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

Example: UCB vs. ϵ-Greedy On 10-armed Bandit

- Average performance of UCB action selection on the 10-armed testbed.
- UCB generally performs better than ϵ-Greedy
- Except in the first k steps, when it selects randomly among the as-yet-untried actions.

$$A_t = \underset{a}{\operatorname{argmax}} \left\{ \hat{Q}_t(a) + c \sqrt{\frac{\log t}{N_t(a)}} \right\}$$



Example: UCB vs. ϵ-Greedy On 10-armed Bandit

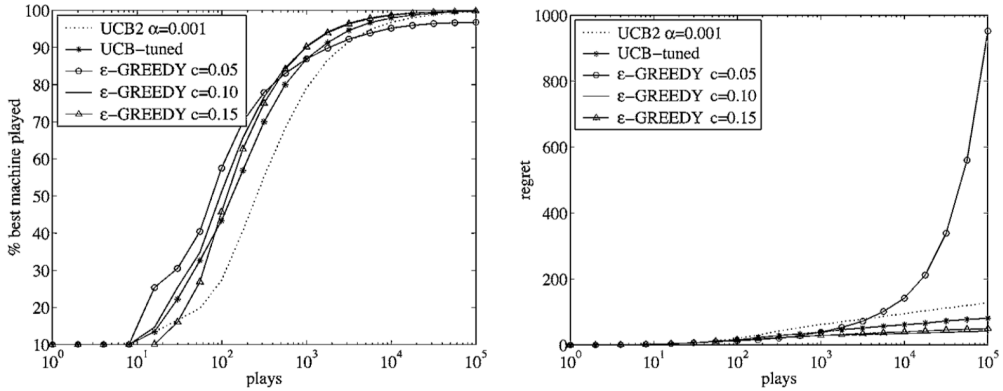


Figure 9. Comparison on distribution 11 (10 machines with parameters 0.9, 0.6, . . . , 0.6).

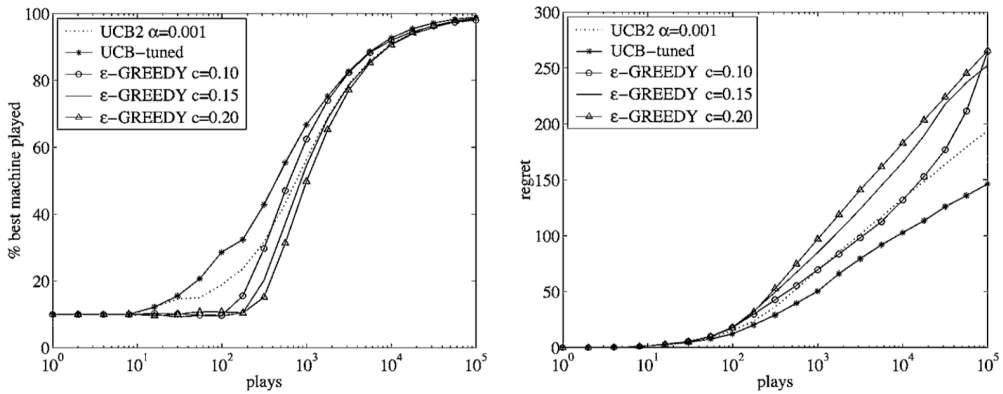


Figure 10. Comparison on distribution 12 (10 machines with parameters 0.9, 0.8, 0.8, 0.8, 0.7, 0.7, 0.7, 0.6, 0.6, 0.6).

- Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer, "Finite-time analysis of the multi-armed bandit problem", Machine Learning, vol.47, no.2-3, pp. 235-256, 2002.

Outline

Thompson Sampling

- The first bandit algorithm proposed by W. R. Thompson in 1933
- For the Bernoulli case with two arms (no theory)
- Almost ignored for nearly 80 years!
- Rediscovered from 2010 & 2011 due to its superior empirical performance over UCB algorithms & others
- Theoretical analysis in 2012 showed that Thompson sampling achieves Lai-Robbins lower bound! (asymptotically optimal)
- Can be difficult to compute analytically from posterior unless the cases of conjugate prior

Beta-Bernoulli Bandit

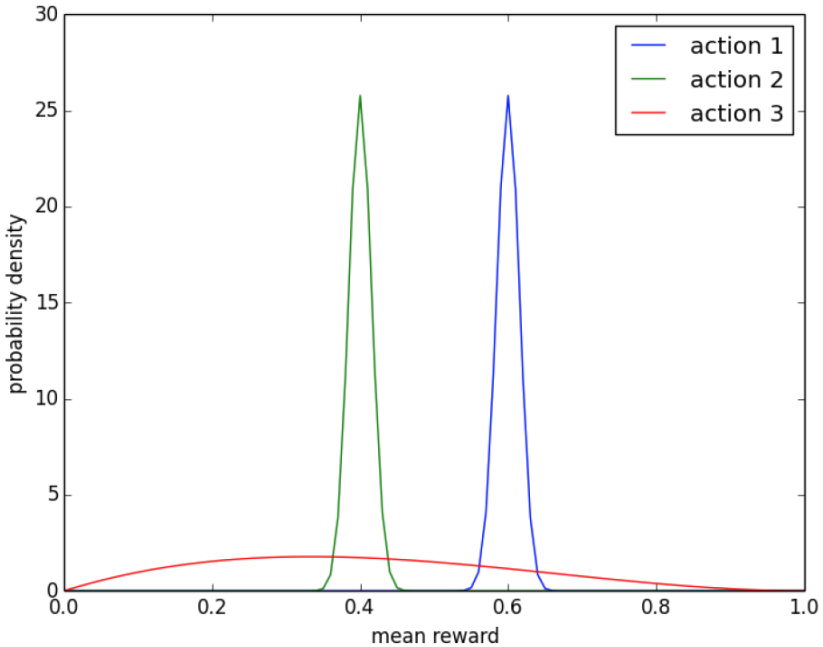
- Stochastic bandits with K actions (arms)
- Action $k \in \{1, \ldots, K\}$ being played produces a reward satisfying Bernoulli distribution Bern(θ_k): reward 1 w.p. θ_k and reward 0 w.p. $1 - \theta_k$.
- θ_k: an action's success probability or mean reward
- The mean rewards $\theta = (\theta_1, \ldots, \theta_K)$ are unknown constants.
- Prior of each θ_k satisfies Beta distribution Beta(α_k, β_k)
- x_t denotes the action selected at time t and r_t denotes the corresponding reward of action x_t
- Each action's posterior distribution is also Beta with parameters updated as follows:

$$(\alpha_k, \beta_k) \leftarrow \begin{cases} (\alpha_k, \beta_k) & \text{if } x_t \neq k \\ (\alpha_k, \beta_k) + (r_t, 1 - r_t) & \text{if } x_t = k. \end{cases}$$

Beta Distribution

- When $\alpha_k = \beta_k = 1$, the Beta distribution is Unif(0,1)
- (α_k, β_k) are also called "pseudo counts"
- Beta distribution Beta(α_k, β_k) has mean $\frac{\alpha_k}{\alpha_k + \beta_k}$
- Beta distribution becomes more concentrated as $\alpha_k + \beta_k$ grows
- Example: Beta$(601, 401)$, Beta$(401, 601)$ and Beta$(2, 3)$.

Action 1 Action 2 Action 3



Greedy vs. 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:
  $(\alpha_{x_t}, \beta_{x_t}) \leftarrow (\alpha_{x_t} + r_t, \beta_{x_t} + 1 - r_t)$
end for

Algorithm 2 BernTS(K, α, β)

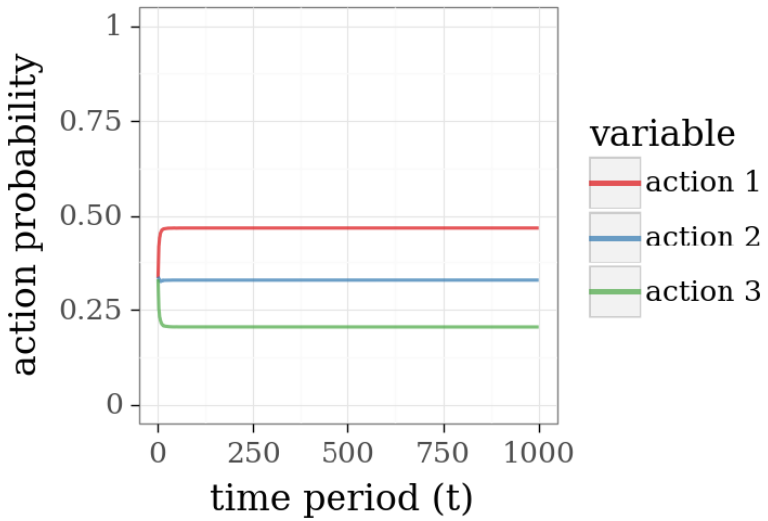
for $t = 1, 2, \ldots$ **do**
  #sample model:
  for $k = 1, \ldots, K$ **do**
    Sample $\hat{\theta}_k \sim \text{beta}(\alpha_k, \beta_k)$
  end for

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

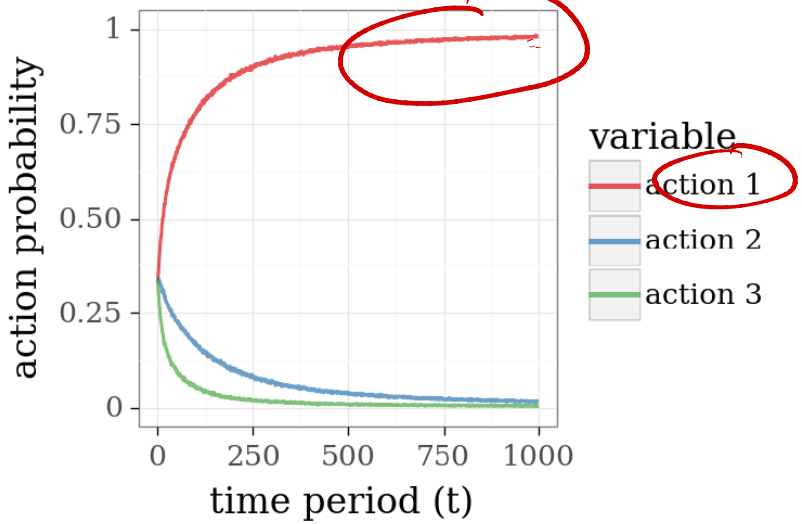
  #update distribution:
  $(\alpha_{x_t}, \beta_{x_t}) \leftarrow (\alpha_{x_t} + r_t, \beta_{x_t} + 1 - r_t)$
end for

Simulation

- Three-armed beta-Bernoulli bandit with mean rewards $\theta_1 = 0.9$, $\theta_2 = 0.8$ and $\theta_3 = 0.7$.
- Prior distributions over each mean reward is uniform.



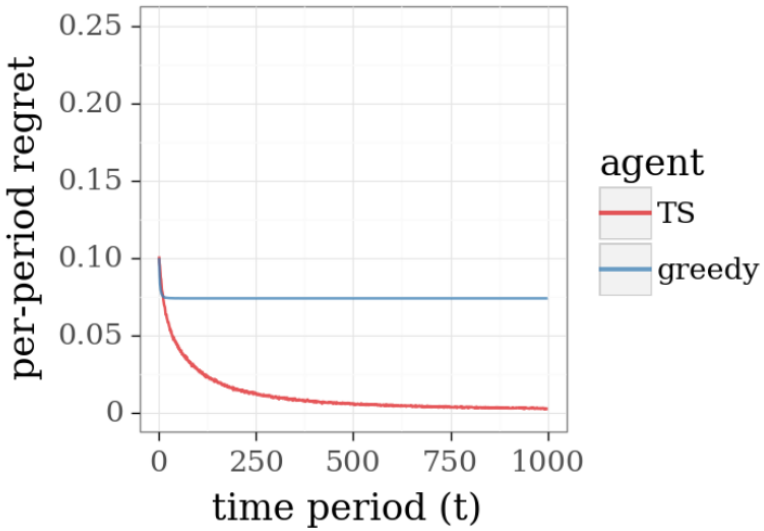
(a) greedy algorithm



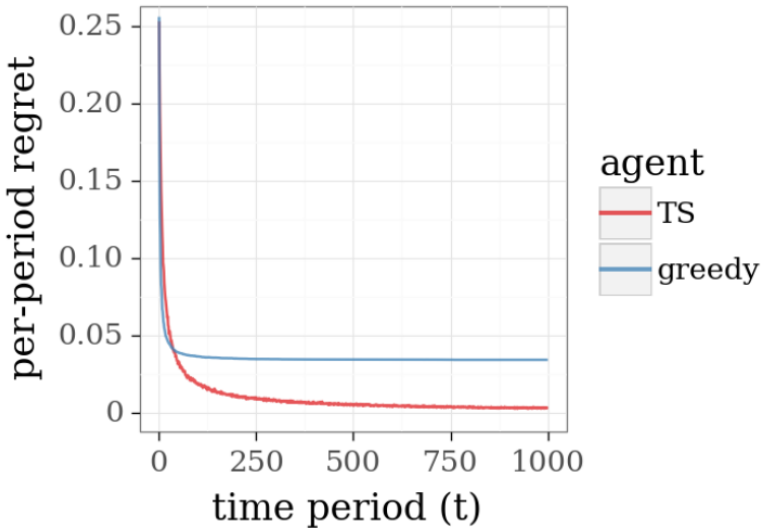
(b) Thompson sampling

Simulation

- Per-period regret: $regret_t(\theta) = \max_k \theta_k - \theta_{x_t}$

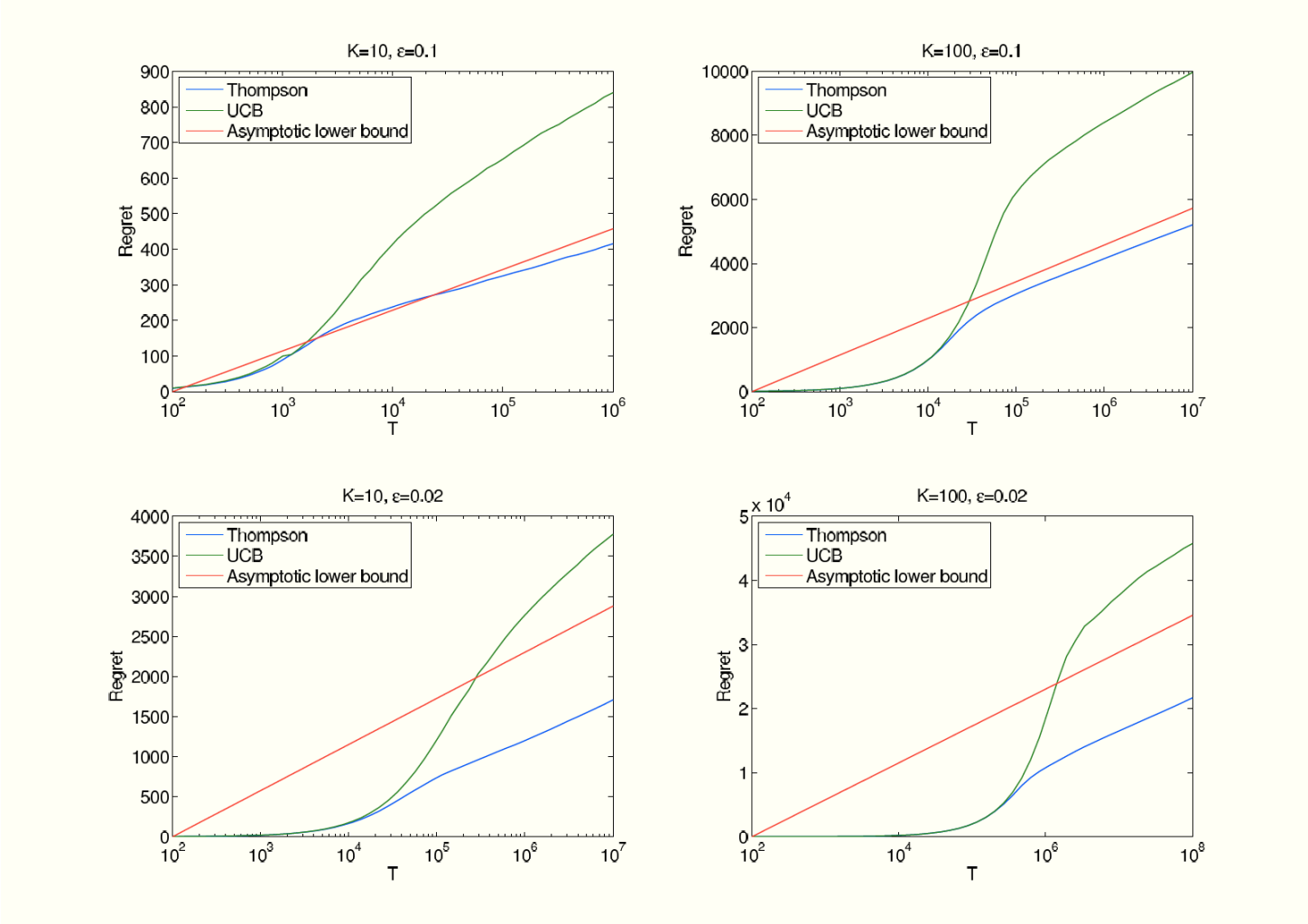


(a) $\theta = (0.9, 0.8, 0.7)$



(b) average over random θ

UCB vs. Thompson Sampling



- O. Chapelle & L. Li, "An Empirical Evaluation of Thompson Sampling", NeuralPS 2011.

Approximate Posterior Sampling

- In many cases exact Bayesian inference is computational intractable
- Four approaches to approximate posterior sampling
  - Gibbs sampling
  - Langevin Monte Carlo
  - Hamiltonian Monte Carlo
  - Sampling from a Laplace approximation

Outline

Gradient Bandit Algorithm

- Let $H_t(a)$ be a learned preference for taking action a

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

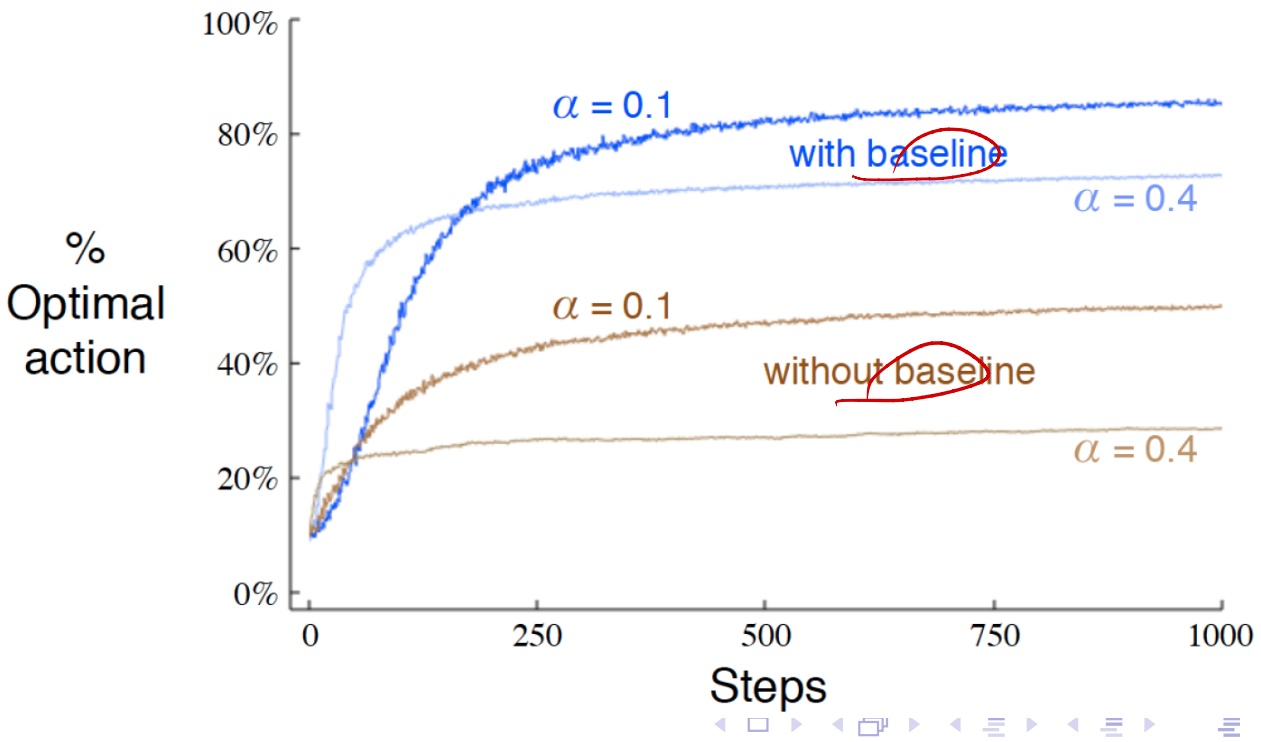
Softmax

Stochastic Gradient Ascent

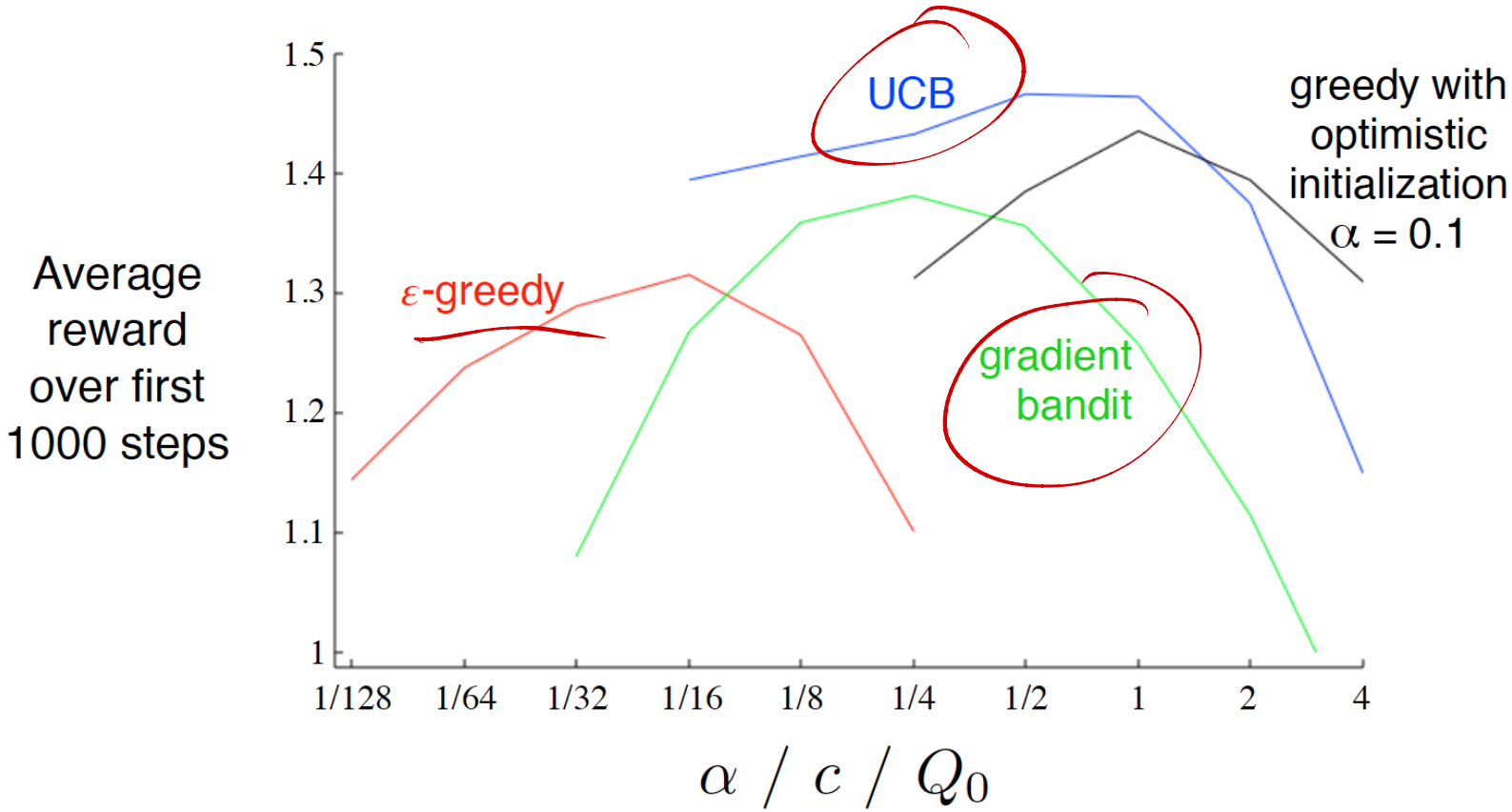
$$H_{t+1}(a) \doteq H_t(a) + \alpha\Big(R_t - \bar{R}_t\Big)\Big(\mathbb{1}\{A_t = a\} - \pi_t(a)\Big), \quad \forall a$$

$$\bar{R}_t \doteq \frac{1}{t}\sum_{i=1}^{t} R_i$$

baseline



Performance Comparison with Other Bandit Algorithms



Why We Study Gradient Bandit Algorithm

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

- A special case of the gradient-based RL algorithms
- Precursor of policy-gradient & actor-critic algorithms in RL

$$\widetilde{\pi}_t(a) \propto e^{\beta H_t(a)}$$

$$\propto e^{\beta_t H_t(a)}$$

Derivation of Gradient Bandit Algorithm

1^0. $\max_w E[f(w)]$ gradient ascent algorithm.

$$w \leftarrow w + \alpha \cdot \nabla_w E[f(w)]$$

α: stepsize.

Stochastic gradient Ascent (SGA) $w \leftarrow w + \alpha \cdot \nabla_w f(w)$

2^0. Now we return to gradient Bandit Algorithm.

MAB: Action (A_t): $x \in \{1, \dots, k\}$

$\max E[R_t]$

$$E[R_t] = E[E[R_t|A_t]] = \sum_x E[R_t|A_t=x] \cdot P(A_t=x)$$

$$= \sum_x q_*(x) \cdot P(A_t=x) = \sum_x q_*(x) \cdot \pi_t(x)$$

$$\pi_t(x) \propto e^{H_t(x)}$$

$\Rightarrow E[R_t]$ is a function of $H_t(x)$. ($w : H_t(x)$)

$\Rightarrow$ SGA: $H_{t+1}(a) = H_t(a) + \alpha \cdot \frac{\partial E[R_t]}{\partial H_t(a)}$, $\forall a \in \{1, \dots, k\}$

Derivation of Gradient Bandit Algorithm

3°. Lemma 1: $\frac{\partial \pi_t(x)}{\partial H_t(a)} = \pi_t(x)\left(\mathbb{1}_{\{x=a\}} - \pi_t(a)\right)$ (Homework)

4°. By Lemma 1, $\frac{\partial r_t(A_t)}{\partial H_t(a)} = \pi_t(A_t)\left(\mathbb{1}_{\{A_t=a\}} - \pi_t(a)\right)$. (*1)

A_t: action in time t.

5°. Baseline: $\mathbb{E}[R_t] = \sum_x q_*(x)\pi_t(x) \Rightarrow \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \sum_x q_*(x)\frac{\partial \pi_t(x)}{\partial H_t(a)}$

Because $\sum_x \pi_t(x) = 1 \Rightarrow \frac{\partial\left(\sum_x \pi_t(x)\right)}{\partial H_t(a)} = 0 \Rightarrow \sum_x \frac{\partial \pi_t(x)}{\partial H_t(a)} = 0$

thus we introduce a Baseline B_t (a scalar that does not depend on x)

$\Rightarrow B_t \cdot \sum_x \frac{\partial \pi_t(x)}{\partial H_t(a)} = 0 \Rightarrow \sum_x B_t \cdot \frac{\partial \pi_t(x)}{\partial H_t(a)} = 0$

$\Rightarrow \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \sum_x q_*(x)\frac{\partial \pi_t(x)}{\partial H_t(a)} - \sum_x B_t \frac{\partial \pi_t(x)}{\partial H_t(a)} = \sum_x (q_*(x) - B_t)\frac{\partial \pi_t(x)}{\partial H_t(a)}$

Derivation of Gradient Bandit Algorithm

$$\Rightarrow \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \sum_x [q_*(x) - B_t] \cdot \frac{\partial \pi_t(x)}{\partial H_t(a)} = \sum_x \pi_t(x) [q_*(x) - B_t] \left(\frac{\partial \pi_t(x)}{\partial H_t(a)} \cdot \frac{1}{\pi_t(x)} \right)$$

Lemma 1

$$= \sum_x \pi_t(x) \cdot [q_*(x) - B_t] \left(\mathbb{1}_{\{x=a\}} - \pi_t(a) \right)$$

$$= \mathbb{E}\left[(q_*(A_t) - B_t) \left(\mathbb{1}_{\{A_t=a\}} - \pi_t(a) \right) \right]$$

$A_t \sim \pi_t$

$q_*(A_t) = \mathbb{E}[R_t | A_t]$

6°. choose $B_t = \overline{R}_t = \frac{1}{t} \sum_{i=1}^{t} R_i$

$$\Rightarrow \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \mathbb{E}\left[(q_*(A_t) - \overline{R}_t) \left(\mathbb{1}_{\{A_t=a\}} - \pi_t(a) \right) \right]$$

7°. Next we will show $= \mathbb{E}\left[(R_t - \overline{R}_t) \left(\mathbb{1}_{\{A_t=a\}} - \pi_t(a) \right) \right]$ ✓ (⇐ 7°)

Proof:

$$\mathbb{E}\left[q_*(A_t) \left(\mathbb{1}_{\{A_t=a\}} - \pi_t(a) \right) \right] = \mathbb{E}\left[R_t \left(\mathbb{1}_{\{A_t=a\}} - \pi_t(a) \right) \right]$$

$$= \mathbb{E}\left[\mathbb{E}\left[R_t \left(\mathbb{1}_{\{A_t=a\}} - \pi_t(a) \right) | A_t \right] \right] \overset{\text{Axiom}}{=} \mathbb{E}\left[R_t \left(\mathbb{1}_{\{A_t=a\}} - \pi_t(a) \right) \right]$$ ✓

Derivation of Gradient Bandit Algorithm

$$8^{\circ}. \quad \frac{\partial E[R_t]}{\partial H_t(a)} = E\left[(R_t - \bar{R}_t)(\mathbb{1}_{\{A_t=a\}} - \pi_t(a))\right]$$

$\Rightarrow$ Gradient Ascent

$$H_{t+1}(a) = H_t(a) + \alpha \cdot \frac{\partial E[R_t]}{\partial H_t(a)}$$

$$= H_t(a) + \alpha E\left[(R_t - \bar{R}_t)(\mathbb{1}_{\{A_t=a\}} - \pi_t(a))\right]$$

$\Rightarrow$ Stochastic Gradient Ascent

$$H_{t+1}(a) = H_t(a) + \alpha \cdot (R_t - \bar{R}_t)(\mathbb{1}_{\{A_t=a\}} - \pi_t(a)),$$

$$\forall\, a \in \{1, \ldots, k\}$$

Derivation of Gradient Bandit Algorithm

q^0. $\max_\theta E_x[f(x)]$, x is a r.v. $\sim P_\theta$ (PMF or PDF)

$$\nabla_\theta E_x[f(x)] = \nabla_\theta \frac{\int p(x) f(x) dx}{\sum_x p(x) f(x)} = \sum_x (\nabla_\theta p(x)) f(x)$$

$$= \sum_x p(x) \cdot \frac{\nabla_\theta p(x)}{p(x)} \cdot f(x) = \sum_x p(x) \cdot \nabla_\theta \log p(x) \cdot f(x)$$

$$= E_x[f(x) \cdot \nabla_\theta \log p(x)]$$

$$\approx \frac{1}{n} \sum_{i=1}^{n} f(x_i) \nabla_\theta (\log p(x_i))$$

Sample mean

Score function

$f(x_i) \cdot \nabla_\theta \log p(x_i)$

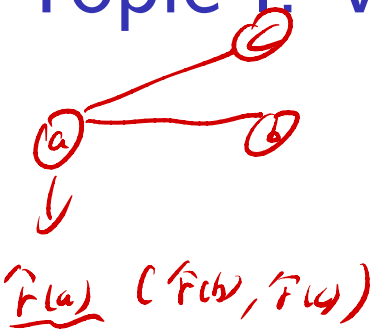
Derivation of Gradient Bandit Algorithm

Derivation of Gradient Bandit Algorithm

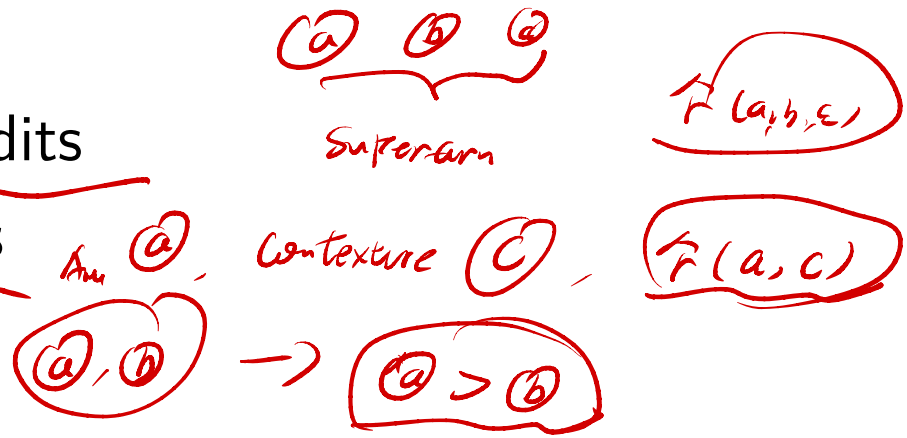
Derivation of Gradient Bandit Algorithm

Outline

Research Topic I: Variants of Feedback



- Graphical Bandits
- Combinatorial Bandits
- Contexture Bandits
- Dueling Bandits
- Batched Bandits



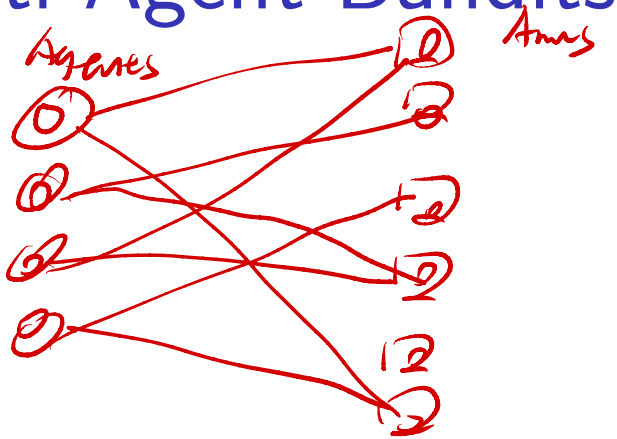
Research Topic II: Functional Bandits

- Linear Bandits
- Kernel Bandits
- Neural Bandits
- Matrix Bandits
- Bayesian Bandits

Research Topic III: Constrained Bandits

- Bandits with hard constraints: knapsack constraints
- Bandits with soft constraints: anytime cumulative constraints & long-term time-average constraints

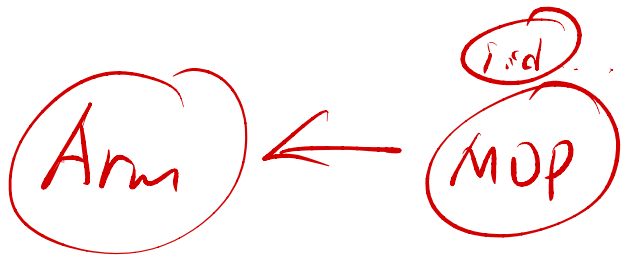
Research Topic IV: Multi-Agent Bandits



- Federated Bandits
- Matching markets between agents and arms

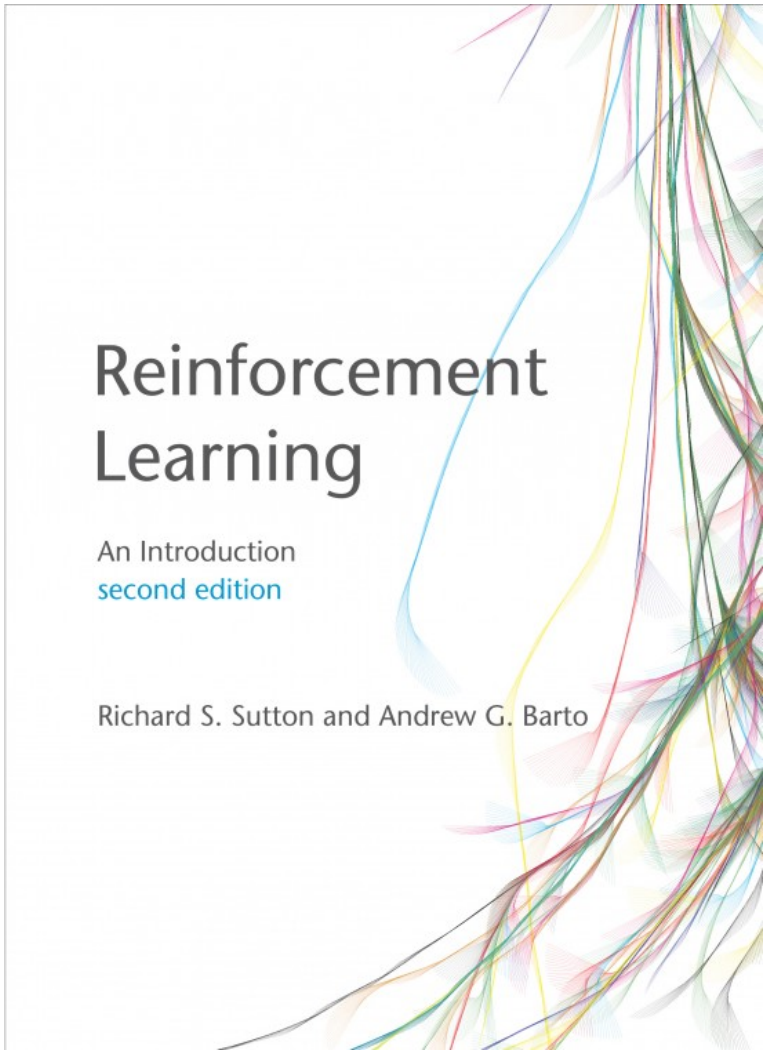
Research Topic V: Additional Structure Information or Setting

- Offline Bandits
- Casual Bandits
- Piecewise-stationary & Non-stationary Bandits
- Adversarial Bandits
- Rested & Restless Bandits



Outline

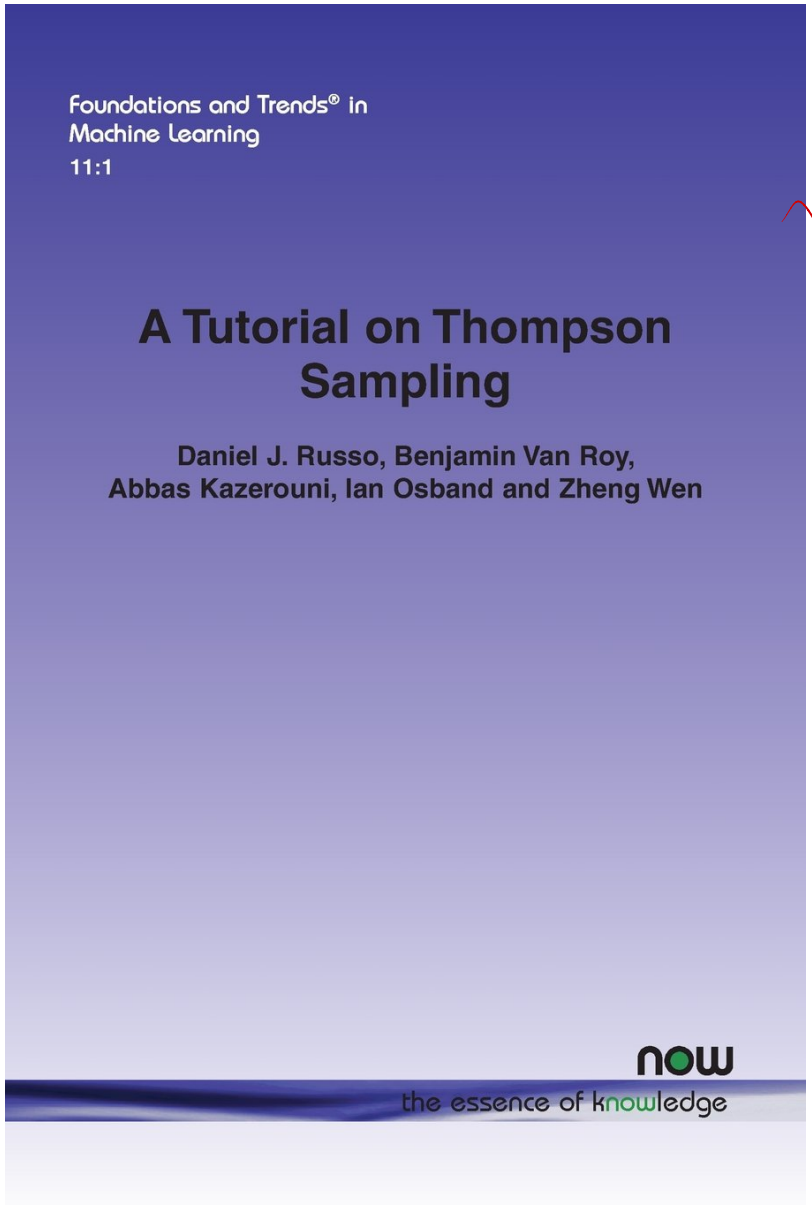
SB



Richard Sutton & Andrew Barto

- Reinforcement Learning: An Introduction
- The MIT Press, 2018.
- Chapter2

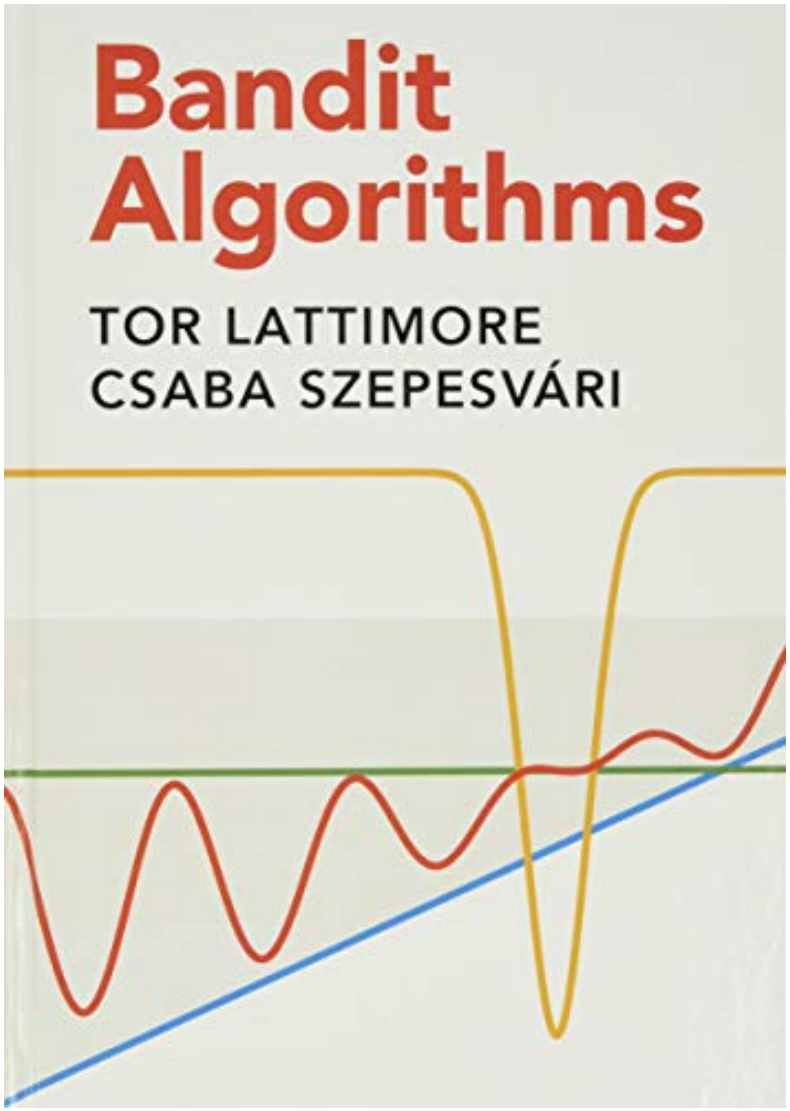
RRKOW



Daniel Russo et al

- A Tutorial on Thompson Sampling
- Foundations and Trends in Machine Learning
- Vol. 11, No. 1, pp. 1-96, 2018

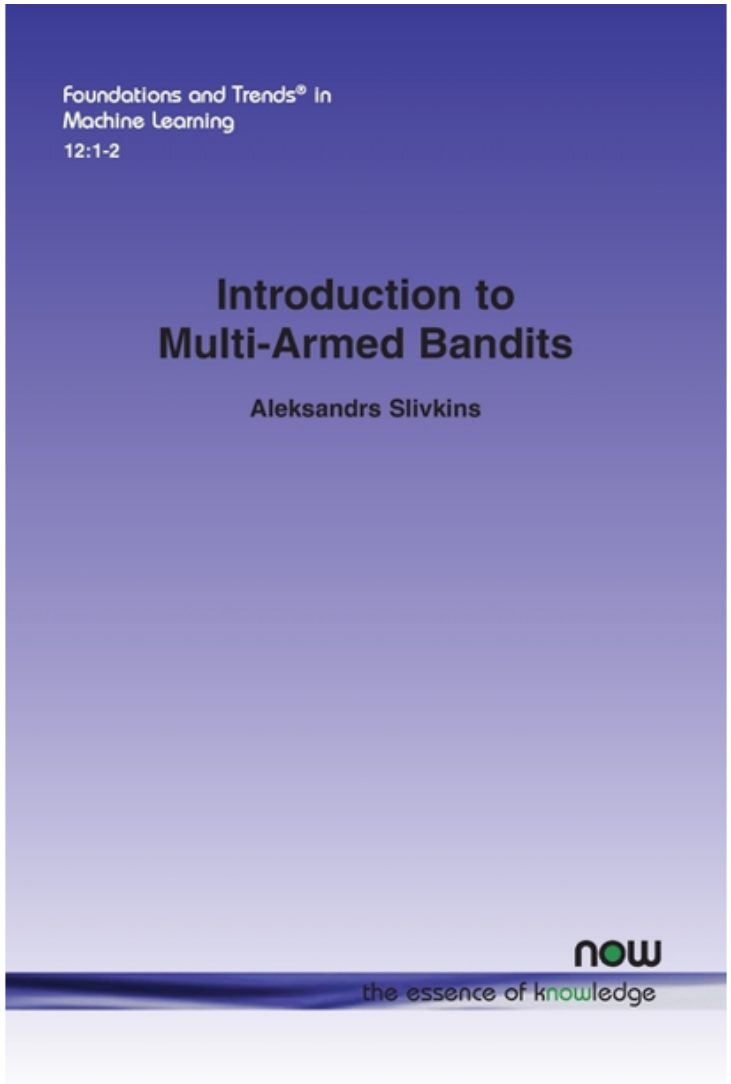
LS



Tor Lattimore & Csaba Szepesvari

- Bandit Algorithms
- Cambridge University Press, 2020.

AS



Aleksandrs Slivkins

- Introduction to Multi-Armed Bandits
- Foundations and Trends in Machine Learning
- Vol. 12, No. 1-2, pp 1-286, 2019