Randomised Algorithms

Lecture 15: Bandit Algorithms

Thomas Sauerwald (tms41@cam.ac.uk)

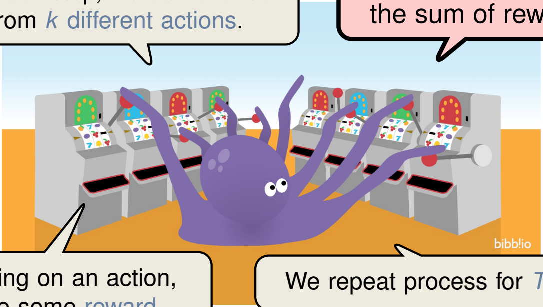
Lent 2022

Outline

Multi-Armed Bandits

At each step, we can choose from k different actions.

How can we maximise the sum of rewards?



After deciding on an action, we receive some reward.

We repeat process for T steps.

Source: Bibblio

Multi-Armed Bandits: make a sequence of decisions under uncertainty.

Bandit Model versus Expert Model

In the Online Learning using Expert setting:

- We have n experts and at each round each expert makes a prediction, which may be correct or wrong
- Our goal is to make a prediction at each round and perform (almost) as good as the best expert.
- Multiplicative-Weight-Update: Each expert suggests a decision which yields to a reward/penalty in $[-1, 1]$ (which is known to us!)

Key Difference: In the Multi-Armed Bandit model, we only observe the cost/reward of the chosen action but not of the other actions!

$\Rightarrow$ Multi-Armed Bandit model is more challenging (and perhaps more realistic?)

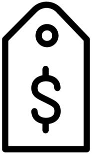
There is a rich interplay between the two models (see EXP3 algorithm later)!

Applications of Multi-Armed Bandits (1/2)

1. News/Ad Selection: When a user visits a news site, a header is presented and the user will click on it or not.
   Goal: maximise the number of clicks.

2. 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.
   Goal: maximise the total profit.

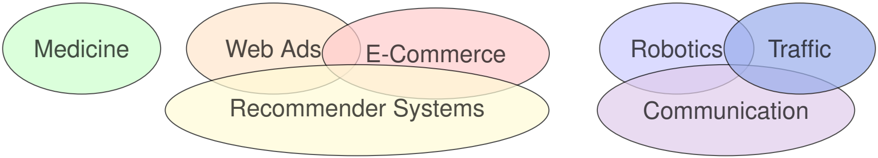
3. Medical Trials: A doctor tries to find an effective treatment against a new virus. Patients arrive one by one, and for each patient the doctor can prescribe one of several possible treatments.
   Goal: cure the maximum number of patients.



Applications of Multi-Armed Bandits (2/2)

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.

Source: Survey by Slivkins



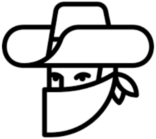
Types of Multi-Armed Bandits Environments

1. **Stochastic (Stationary) Bandits:** Environment generates random reward to each action that is specific to that action and independent of the previous actions and rewards.

2. **Bayesian Bandits:** Use a prior probability measure on the reward distribution that reflects our initial belief. With every action, the learner can update the prior by a new posterior distribution.

3. **Adversarial Bandits:** No assumption on how rewards are generated, apart from that rewards are determined without knowing the learner's action.

4. **Contextual Bandits:** We have access to additional information that may help predicting the quality of the actions at each time (e.g., demographical information or preferences of users).



How The New York Times is Experimenting with Recommendation Algorithms

M

NYT Open

Algorithmic curation at The Times is used in designated parts of our website and apps.

Anna Coenen Follow

Oct 17, 2019 · 6 min read

A contextual recommendation approach

One recommendation approach we have taken uses a class of algorithms called contextual multi-armed bandits. Contextual bandits learn over time how people engage with particular articles. They then recommend articles that they predict will garner higher engagement from readers. The *contextual* part means that these bandits can use additional information to get a better estimate of how engaging an article might be to a particular reader. For example, they can take into account a reader's geographical region (like country or state) or reading history to decide if a particular article would be relevant to that reader.

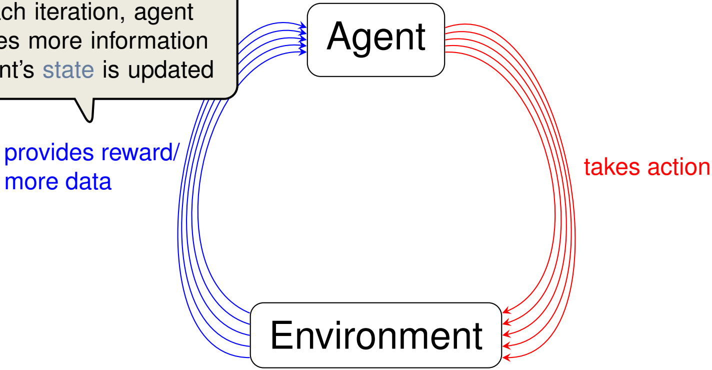
["recommended": "article B"; "reader state": "Texas", "clicked": "yes"]
["recommended": "article A"; "reader state": "New York", "clicked": "yes"]
["recommended": "article B"; "reader state": "New York", "clicked": "no"]
["recommended": "article B"; "reader state": "California", "clicked": "no"]
["recommended": "article A"; "reader state": "New York", "clicked": "no"]

Once the bandit has been trained on the initial data, it might suggest Article A, Article B or a new article, C, for a new reader from New York. The bandit would be most likely to recommend Article A because the article had the highest click-through rate with New York readers in the past. With some smaller probability, it might also try showing Article C, because it doesn't yet know how engaging it is and needs to generate some data to learn about it.

Online Algorithm/Reinforcement Learning Framework

Exploration vs. Exploitation

In each iteration, agent receives more information $\Rightarrow$ agent's state is updated

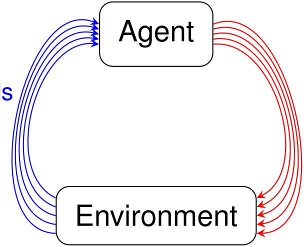


Iteration:

Outline

The Language of Bandits

receives rewards
r_1, r_2, r_3, r_4, r_5

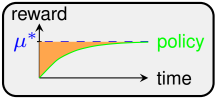


takes action sequence (policy)
a_1, a_2, a_3, a_4, a_5

Iteration: $1, 2, 3, 4, 5$

- Let a_t be the action and r_t be the (unknown) reward at step t
- Let $\mu(a) := \mathbf{E}[\, r_t \mid a_t = a \,]$ be the mean reward given action a, and $\mu^* = \max_a \mu(a)$ be the maximal mean reward and $a^* = \operatorname{argmax}_a \mu(a)$.
- The **(cumulative) regret** of a policy $\pi = (a_1, a_2, \ldots)$ is

$$R_T(\pi) = T \cdot \mu^* - \sum_{t=1}^{T} \mu(a_t).$$



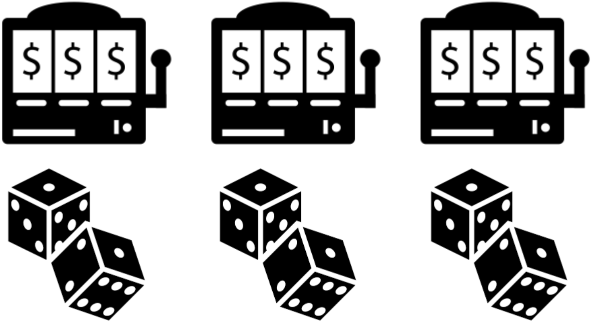
Comparing against the mean-optimal strategy ("best-arm benchmark")

This sum depends on the policy π and horizon T, but it is deterministic.

Stochastic (Bernoulli) Bandits

- Consider the time-horizon $1, 2, \ldots, T$
- We have k different actions (arms) at each step
- Every reward is a binary random variable with unknown probability

This is also known as Bernoulli Bandits



Regret in Bernoulli Bandits: Example

Let $k = 3$

t	Available Actions	Reward	Total (Realised) Reward
1	$1, 2, 3$	0	0
2	$1, 2, 3$	1	1
3	$1, 2, 3$	1	2
4	$1, 2, 3$	0	2
5	$1, 2, 3$	1	3
6	$1, 2, 3$	0	3
7	$1, 2, 3$	1	4
8	$1, 2, 3$	0	4
9	$1, 2, 3$	1	5
10	$1, 2, 3$	1	6

Exercise: Assume $\mu(1) = 0.4$, $\mu(2) = 0.5$, $\mu(3) = 0.7$. What is the regret?

1. Compute maximal mean reward $T \cdot \mu^*$
2. Compute mean reward of used policy $(1, 2, 2, 2, 3, 3, 2, 2, 3, 3)$

Regret in Bernoulli Bandits: Example

Let $k = 3$

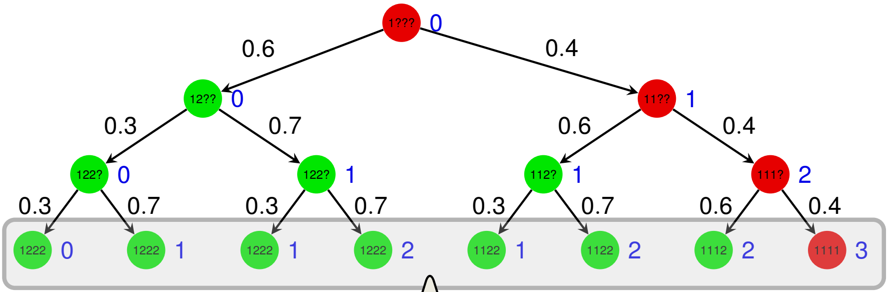
t	Available Actions	Reward	Total (Realised) Reward
1	1, 2, 3	0	0
2	1, 2, 3	1	1
3	1, 2, 3	1	2
4	1, 2, 3	0	2
5	1, 2, 3	1	3
6	1, 2, 3	0	3
7	1, 2, 3	1	4
8	1, 2, 3	0	4
9	1, 2, 3	1	5
10	1, 2, 3	1	6

1. Maximal mean reward is $T \cdot \mu^* = 10 \cdot 0.7 = 7$
2. Mean reward of our policy is $1 \cdot 0.4 + 5 \cdot 0.5 + 4 \cdot 0.7 = 5.7$

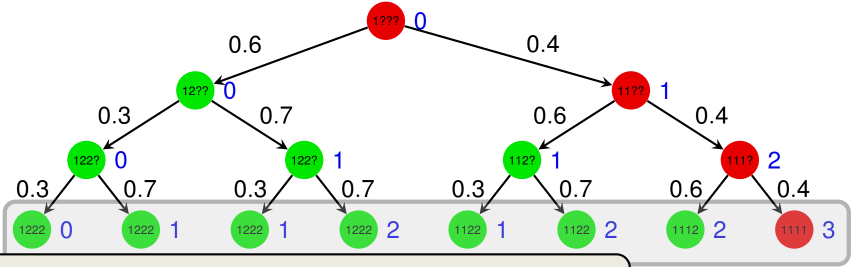
$\Rightarrow$ Cumulative Regret is $7 - 5.7 = 1.3$

Question: Why does regret involve mean rewards and not realised rewards?

- Consider two Bernoulli bandits with probabilities 0.4 and 0.7
- Our policy is: start with first arm and switch to second (and stay with it) as soon as we don't get a reward from first arm
- Consider $t = 3$ and the realised reward
- Expected realised reward is the (weighted) average over the rewards of the 8 leafs

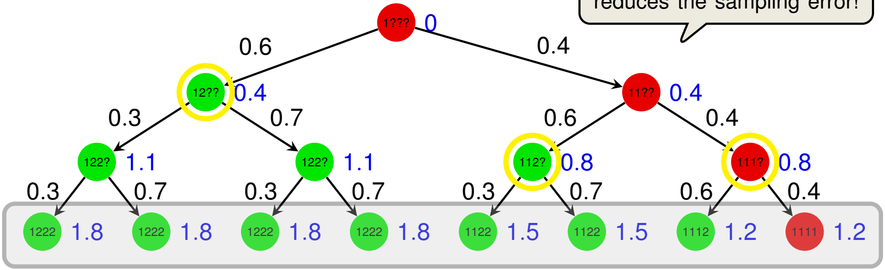


Expected Reward is: $0.6 \cdot 0.3 \cdot 0.3 \cdot 0 + 0.6 \cdot 0.3 \cdot 0.7 \cdot 1 + \ldots$



- Let us change the reward calculation to mean reward!
- The expected reward is now easier to compute:
  $0.6 \cdot 1.8 + 0.4 \cdot 0.6 \cdot 1.5 + 0.4 \cdot 0.4 \cdot 1.2 = 1.632$

Using mean reward reduces the sampling error!



A Simple Heuristic

Algorithm 1: Greedy

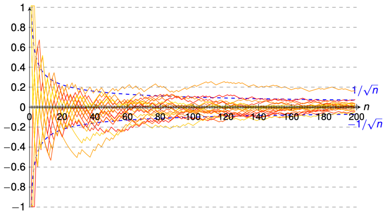
- Idea: Choose the arm with the highest average realised reward so far

1. That is, for every action a and step t, we compute

$$Q_t(a) = \frac{\text{sum of rewards when } a \text{ taken until time } t}{\text{number of times } a \text{ taken until time } t} = \frac{\sum_{i=1}^{t-1} \mathbf{1}_{a_i=a} \cdot r_i}{\sum_{i=1}^{t-1} \mathbf{1}_{a_i=a}}$$

2. Then choose the action $a_t = \operatorname{argmax}_a Q_t(a)$

This is a general method called action-value method: guide decisions by estimating values of actions



Exercise: Do you think this is a good strategy?

Regret in Bernoulli Bandits: Greedy on Earlier Example

Let $k = 3$ and $\mu(1) = 0.4$, $\mu(2) = 0.5$, $\mu(3) = 0.7$.

t	Available Actions	Reward	Realised Reward	Mean Reward
1	1, 2, 3	0	0	0.4
2	1, 2, 3	1	1	0.9
3	1, 2, 3	1	2	1.4
4	1, 2, 3	0	2	1.9
5	1, 2, 3	1	3	2.4
6	1, 2, 3	0	3	2.9
7	1, 2, 3	0	3	3.4
8	1, 2, 3	0	3	3.9
9	1, 2, 3	1	4	4.4
10	1, 2, 3	0	4	4.9

1. Greedy will in the long run achieve reward of $T \cdot 0.5$
2. Greedy will never try action 3, which is better! **Not enough exploration!**

Improving Greedy

Algorithm 2: ϵ-Greedy

- Idea: With probability $\epsilon \in (0,1)$ pick an action uniformly at random, otherwise perform Greedy

$\Rightarrow$ Since every action is sampled infinitely often, we have

$$\lim_{t\to\infty} Q_t(a) = \lim_{t\to\infty} \frac{\text{sum of rewards when } a \text{ taken until time } t}{\text{number of times } a \text{ taken until time } t} = \mu(a).$$

Hence the algorithm will eventually "learn" optimal policy and the regret is small.

How should we choose ϵ in order to minimise the regret?

Regret in Bernoulli Bandits: Example of ϵ-Greedy

$k = 3$, $\epsilon = 1/2$ and $\mu(1) = 0.4$, $\mu(2) = 0.5$, $\mu(3) = 0.7$

t	Available Actions	Reward	Realised Reward	Mean Reward
1	$1, 2, 3$	0	0	0.4
2	$1, 2, 3$	1	1	0.9
3	$1, 2, 3$	0	1	1.3
4	$1, 2, 3$	0	1	1.8
5	$1, 2, 3$	0	1	2.5
6	$1, 2, 3$	0	1	3
7	$1, 2, 3$	0	1	3.5
8	$1, 2, 3$	0	1	3.9
9	$1, 2, 3$	1	2	4.6
10	$1, 2, 3$	1	3	5.3

1. ϵ-Greedy may take a lot of sub-optimal actions at the beginning
2. However, it explores all actions often enough!

Experimental Results: Greedy and ϵ-Greedy (1/2)

To roughly assess the relative effectiveness of the greedy and ε-greedy action-value methods, we compared them numerically on a suite of test problems. This was a set of 2000 randomly generated k-armed bandit problems with $k = 10$. For each bandit problem, such as the one shown in Figure 2.1, the action values, $q_*(a)$, $a = 1, \ldots, 10$,

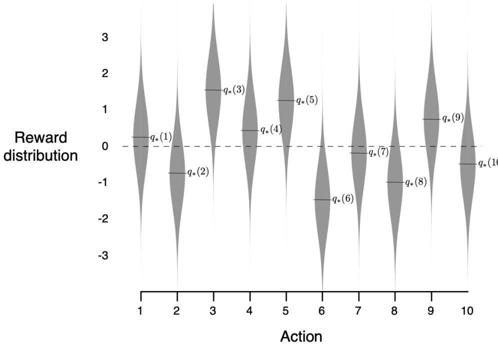
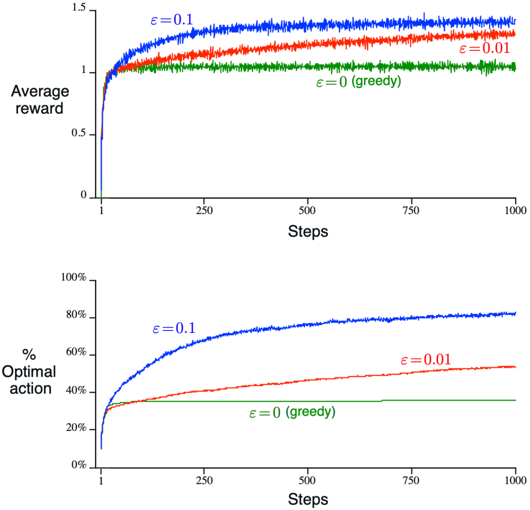


Figure 2.1: An example bandit problem from the 10-armed testbed. The true value $q_*(a)$ of each of the ten actions was selected according to a normal distribution with mean zero and unit variance, and then the actual rewards were selected according to a mean $q_*(a)$, unit-variance normal distribution, as suggested by these gray distributions.

Source: Sutton and Barto

Experimental Results: Greedy and ϵ-Greedy (2/2)



Source: Sutton and Barto

Intuition: How to Pick ϵ

1. If $\epsilon_t = \epsilon$ is any constant $\in (0,1)$, then:

$$\mathbf{P}\left[\, a_t \neq a^* \,\right] \approx \epsilon.$$

$\Rightarrow$ Even if we have learned optimal action, regret may grow linear in T:

$$R_T(\mu) = T \cdot \mu^* - \sum_{t=1}^{T} \mu(a_t) \approx \sum_{t=1}^{T} \epsilon = \Omega(T).$$

2. If $\epsilon_t = 1/t$, then:

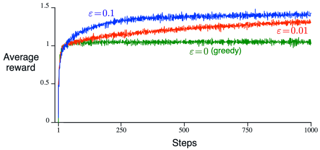
$$\mathbf{P}\left[\, a_t \neq a^* \,\right] \approx \epsilon_t = 1/t.$$

$\Rightarrow$ Hence we may hope regret grows logarithmic in T, i.e.,

$$R_T(\mu) \approx \sum_{t=1}^{T} \epsilon_t = O(\log T).$$

Exercise: What happens if $\epsilon_t = 1/t^2$?

Summary so far...



Source: Sutton and Barto

For $\epsilon_t = \Theta(1/t)$, ϵ-Greedy achieves a regret of $O(\log T)$.

- This can be shown formally (under some mild technical assumptions)

  [Auer, Cesa-Binchi and Fischer; "Finite-Time Analysis of the Multiarmed Bandit Problem", 2002]
- Downside: ϵ-Greedy algorithm does not adjust its strategy based on the experienced reward (it may take arms with no reward too often)

Ideas for Improvements:

- In an exploration step, sample non-uniformly
- Blend exploration and exploitation by maintaining for each arm an **upper confidence bound** for the mean reward

Towards the UCB Algorithm

Question: How close are $Q_t(a)$ (the empirical estimate) and $\mu(a)$?

This makes only sense if $\Delta_t(a)$ becomes small, but as $t \to \infty$, we get more data and indeed $\Delta_t(a) \to 0$.

Idea of the Upper Confidence Bound Algortihm

1. Suppose for every action a, there is a bound $\Delta_t(a) \geq 0$ such that:

$$|Q_t(a) - \mu(a)| \leq \Delta_t(a) \qquad \Rightarrow \qquad \mu(a) \leq Q_t(a) + \Delta_t(a).$$

2. Now pick "greedily" the arm with largest $\widetilde{\mu}_t(a) := Q_t(a) + \Delta_t(a)$

Principle of Optimism in the Face of Uncertainty:

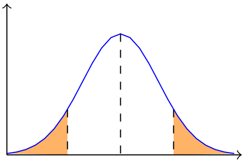
- For each action, construct an optimistic guess for the expected reward
- At each step, we pick the action with the largest guess
- If that action turned out to be "too optimistic", then next guess will be lower

$\Rightarrow$ generally prefer arms with high empirical reward and/or high uncertainty

Chernoff Bounds

Question: How close are $Q_t(a)$ (the empirical estimate) and $\mu(a)$?

Want Confidence-Bound like $\mathbf{P}\left[\,|Q_t(a) - \mu(a)| \leq \Delta_t(a)\,\right] \geq 1 - \delta$



Make sure Confidence Bound always holds: $\delta = 2t^{-2}$.

$\lambda(a) = \sqrt{n_t(a) \cdot \log(t)}$

$\Delta_t(a) = \sqrt{\frac{\log(t)}{n_t(a)}}$

Chernoff Bound ("Nicer Version", Lecture 2/3 slide 16)

Let $X_1, \ldots, X_n$ be n independent Bernoulli random variables. Let $X := \sum_{i=1}^{n} X_i$ and $\mu = \mathbf{E}[\,X\,]$. Then, for any $\lambda \geq 0$,

$$\mathbf{P}\left[\,|X - \mu| \geq \lambda\,\right] \leq 2 \cdot \exp\left(-\frac{2\lambda^2}{n}\right).$$

In our application, $n = n_t(a) := \sum_{i=1}^{t} \mathbf{1}_{a_i = a}$, $X = Q_t(a) \cdot n$.

The UCB Algorithm

Algorithm 3: UCB Algorithm

Initialisation: Let $n_1(a) = 0$ and $Q_1(a) = 0$ for all actions a

Execute: For $t = 1, 2, \ldots, T$:

- Take a that maximises $\widetilde{\mu}(a) = Q_t(a) + \sqrt{\frac{\log(t)}{n_t(a)}}$ and receive reward r_t
- Update:

$$n_{t+1}(a) \leftarrow n_t(a) + 1$$

$$Q_{t+1}(a) \leftarrow \frac{n_t(a)Q_t(a) + r_t}{n_t(a) + 1}$$

Smart Update – no extra memory or computations needed!

No parameters needed (like learning rate)!

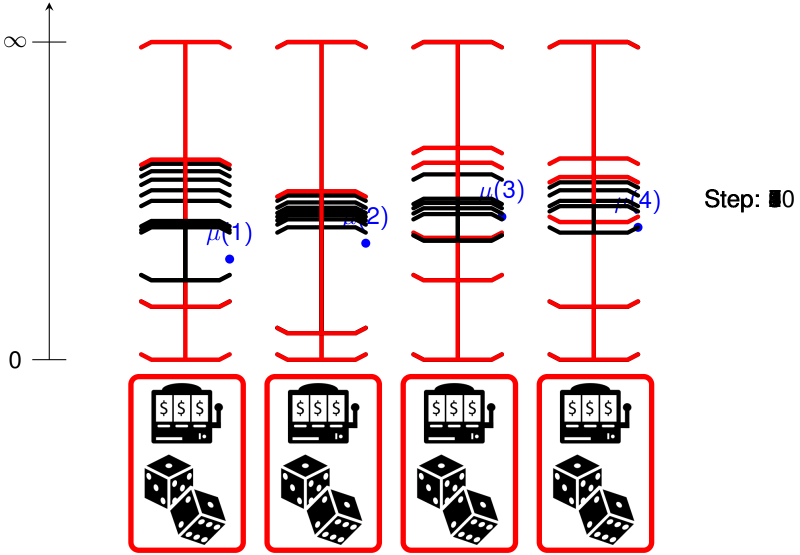
- Recall our high-confidence upper bound: $|Q_t(a) - \mu(a)| \leq \Delta_t(a) = \sqrt{\frac{\log(t)}{n_t(a)}}$.

$\Rightarrow$ To allow us to identify the optimal arm a^*, we need $n_t(a) \approx \log(t)$

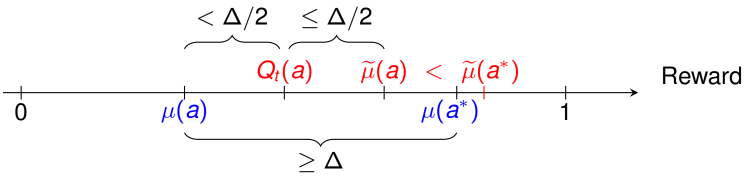
$\Rightarrow$ Hence any sub-optimal arm $a \neq a^*$ will be only taken $\log(T)$ times.

UCB-Algo takes sub-optimal actions only at a logarithmic rate!

Example 1: Illustration of UCB (simplified)



Intuition: How UCB avoids sub-optimal arms



- Let a be a sub-optimal action with $\mu(a) \leq \mu(a^*) - \Delta$
- Optimism: For any action, in particular a^*, we have with probability $1 - \delta_t$,

$$\widetilde{\mu}(a^*) = Q_t(a^*) + \Delta_t(a^*) \geq (\mu(a^*) - \Delta_t(a^*)) + \Delta_t(a^*) = \mu(a^*).$$

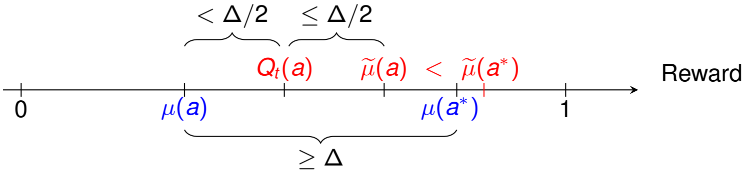
- Let's upper bound $\widetilde{\mu}(a)$, with probability $1 - \delta_t$:

$$\widetilde{\mu}(a) = Q_t(a) + \Delta_t(a) \leq (\mu(a) + \Delta_t(a)) + \Delta_t(a)$$
$$= \mu(a) + 2 \cdot \sqrt{\frac{\log(t)}{n_t(a)}}.$$

- If $n_t(a) > \frac{4\log(t)}{\Delta}$, then $\widetilde{\mu}(a) < \mu(a) + 2 \cdot \Delta/2 = \mu(a) + \Delta$

$\Rightarrow$ $\widetilde{\mu}(a) < \widetilde{\mu}(a^*)$, meaning UCB will **not** take action a (w.p. $1 - \delta_t$)

Intuition: How UCB avoids sub-optimal arms



$\Rightarrow$ $\widetilde{\mu}(a) < \widetilde{\mu}(a^*)$, meaning UCB will **not** take action a (w.p. $1 - \delta_t$)

- Using $R_T = \sum_{a:\ \mu(a)<\mu(a^*)} n_T(a) \cdot (\mu(a^*) - \mu(a))$ one can derive:

Performance of UCB

For any $T \geq 1$, the regret satisfies:

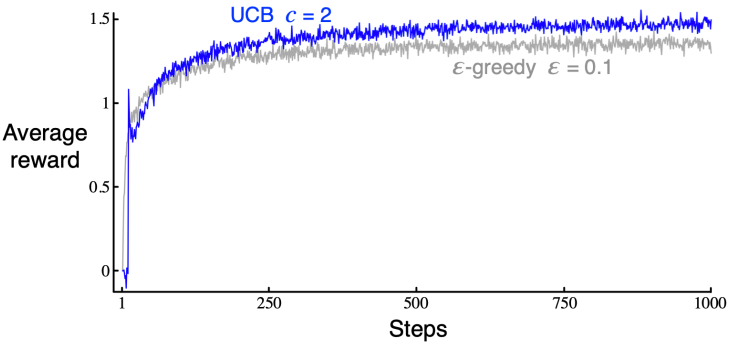
$$R_T \leq \sum_{a:\ \mu(a)<\mu(a^*)} \left(\frac{4 \log(T)}{\mu(a^*) - \mu(a)} + 8(\mu(a^*) - \mu(a)) \right) \approx O(\log(T)).$$

One can also prove a lower bound of $\Omega(\log(T))$ for any algorithm!

contribution from rounds where $n_t(a)$ is small and...

...rounds where $n_t(a)$ is large.

Experimental Results: ϵ-Greedy and UCB



Source: Sutton and Barto

Notes:

- This is the same bandit setting as on slides 20–21
- The UCB algorithm above uses $\Delta_t(a) = 2\sqrt{\frac{\log(t)}{n_t(a)}}$

Thank you and Best Wishes for the Exam!

If you have any questions, comments or feedback, please send an email to `tms41@cam.ac.uk`

Outline

Introduction

Stochastic Bandits

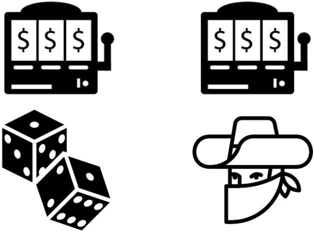
Outlook: Adversarial Bandits (non-examinable)

Why Adversarial Bandits?

Stochastic Bandits

- Rewards of each arm are i.i.d. samples in $[0, 1]$
- distribution is specific to each arm but is time-invariant (stationarity)

Nice model, but assumptions a bit questionable in real-world applications!



Adversarial Bandits

- rewards are in the interval $[0, 1]$
- all rewards must be determined before action is taken

Very weak assumptions $\rightsquigarrow$ powerful model!

Bandits with Full Information: Online Learning using Experts

The Multiplicative Weights Algorithm (MWA)

Initialization: Fix $\delta \leq 1/2$. For every $i \in [n]$, let $w_i^{(1)} := 1$

Update: For $t = 1, 2, \ldots, T$:

- Choose expert i with prop. proportional to $w_i^{(t)}$.
- Observe the costs of all n experts in round t, $r^{(t)} \in [-1, 1]$
- For every expert i, update its weight by:

$$w_i^{(t+1)} = (1 - \delta r_i^{(t)}) w_i^{(t)} \approx \exp\left(-\delta r_i^{(t)}\right) w_i^{(t)}$$

Hence $w_i^{(t+1)} = \exp\left(-\delta \sum_{i=1}^{t} r_i^{(t)}\right)$.

- MWA samples with a proportional that is exponential in the performance of each expert
- We would like to apply the same idea to the Bandit setting
- Problem: In the bandit-setting, we only observe the cost (reward) of the **taken action**

The EXP3-Algorithm

EXP3 $=$ Exponential-weight algorithm for Exploration and Exploitation

The EXP3-Algorithm

Initialization: Fix $\gamma \in (0,1)$. Let $w_1(a) := 1$ for each of the k actions

For $t = 1, 2, \ldots, T$:

action is sampled proportional to weights!

- Define:

$$p_t(a) := \frac{w_t(a)}{\sum_{a'} w_t(a')},$$

  and choose action i with probability $p_t(a)$.

- Observe the reward $r_t(a) \in [0,1]$
- Update weights:

$$w_{t+1}(a) = w_t(a) \cdot \exp\left(\frac{\gamma}{k} \cdot \frac{r_t(a)}{p_t(a)}\right)$$

$$w_{t+1}(a') = w_t(a) \cdot \exp(0) \qquad \text{for all } a \neq a'.$$

The expected change in the exponent is:

$$p_t(a) \cdot \frac{\gamma}{k} \cdot \frac{r_t(a)}{p_t(a)} + (1 - p_t(a)) \cdot 0 = \frac{\gamma}{k} \cdot r_t(a).$$

EXP3-algorithm tries to emulate the full information (expert) setting!

Analysis of EXP3-Algorithm

In the full-information (expert setting), we could achieve $R_T = O(\sqrt{T \log(k)})$!

Performance of EXP3-Algorithm (Auer, Cesa-Bianchi, Freund, Shapire 2002)

For any $T \geq 1$, the expected regret of EXP3 with $\gamma = \sqrt{\frac{\log(k)}{kT}}$ satisfies

$$R_T \leq 2\sqrt{T \cdot k \log(k)}.$$

There is a nearly matching lower bound for any k, T:

$$R_T = \Omega\left(\sqrt{T \cdot k}\right).$$

Remarks:

- Recall: regret-bound compares against the best-arm benchmark
- The analysis is similar to MWA, but more complicated.
- Regret-bound is still sub-linear in T (which is impressive!),
  but it is much higher than in case of stochastic bandits or expert setting
  (recall we are making no assumption on how rewards are determined!)