Lecture 4 - Variants of Bandit Problems

Pratik Gajane

September 19, 2022

Eindhoven University of Technology

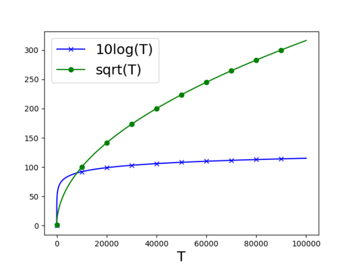
Introduction

Clarification : About the base of log terms

- Base of log terms is mostly not important in this course.
- We are concerned with the leading terms i.e., whether the regret bound is in terms of T or $\sqrt{T}$ or $\log T$ and not with constants.

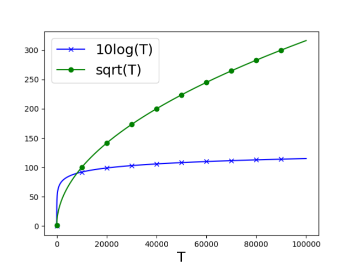
Clarification : About the base of log terms

- Base of log terms is mostly not important in this course.
- We are concerned with the leading terms i.e., whether the regret bound is in terms of T or $\sqrt{T}$ or $\log T$ and not with constants.
- For this course, a regret bound of $logT$ is not *better* than $10logT$ but a regret bound of $10logT$ is *better* than $\sqrt{T}$.



Clarification : About the base of log terms

- Base of log terms is mostly not important in this course.
- We are concerned with the leading terms i.e., whether the regret bound is in terms of T or $\sqrt{T}$ or $\log T$ and not with constants.
- For this course, a regret bound of $logT$ is not *better* than $10logT$ but a regret bound of $10logT$ is *better* than $\sqrt{T}$.



- You can convert the base of a *log* from e to 2 or 10 (and vice versa) just with an extra multiplicative constant (see here).
- So the base only affects the constant in the results, and hence it is mostly not important.

A Quick Recap of Lecture 1, 2 and 3

- Lecture 1 : Introduction to reinforcement learning and its basic elements.
- Lecture 2 : UCB for stationary stochastic bandits and its regret bound. Frequentist perspective.
- Lecture 3 : Thompson sampling for stationary stochastic bandits and its regret bound. Bayseian perspective.

A Quick Recap of Lecture 1, 2 and 3

- Lecture 1 : Introduction to reinforcement learning and its basic elements.
- Lecture 2 : UCB for stationary stochastic bandits and its regret bound. Frequentist perspective.
- Lecture 3 : Thompson sampling for stationary stochastic bandits and its regret bound. Bayseian perspective.

A Quick Recap of Lecture 1, 2 and 3

- Lecture 1 : Introduction to reinforcement learning and its basic elements.
- Lecture 2 : UCB for stationary stochastic bandits and its regret bound. Frequentist perspective.
- Lecture 3 : Thompson sampling for stationary stochastic bandits and its regret bound. Bayseian perspective.

Stationary Stochastic Bandits

- Number of arms $= K$.
- Reward for arm $a \sim X_a$ with mean μ_a.
- $X_1, X_2, \ldots X_K$ are unknown stationary distributions.
- At each time step $t = 1, \ldots, T$, the agent,
  - chooses an arm $i(t)$, and
  - receives a numerical reward $r(t) \sim X_{i(t)}$.

Assumptions in our Bandits Model so far

Stationary Stochastic Bandits

- Number of arms $= K$
- Reward for arm $a \sim X_a$ with mean μ_a.
- $X_1, X_2, \ldots X_K$ are unknown stationary distributions.
- At each time step $t = 1, \ldots, T$, the agent,
  - chooses an arm $i(t)$, and
  - receives a numerical reward $r(t) \sim X_{i(t)}$.

Assumptions in our bandit model so far...

Assumptions in our Bandits Model so far

Stationary Stochastic Bandits

- Number of arms = K
- Reward for arm $a \sim X_a$ with mean μ_a.
- $X_1, X_2, \ldots X_K$ are unknown stationary distributions.
- At each time step $t = 1, \ldots, T$, the agent,
  - chooses an arm $i(t)$, and
  - receives a numerical reward $r(t) \sim X_{i(t)}$.

Assumptions in our bandit model so far...

- Reward distributions are stationary.

Assumptions in our Bandits Model so far

Stationary Stochastic Bandits

- Number of arms = K
- Reward for arm $a \sim X_a$ with mean μ_a.
- $X_1, X_2, \ldots X_K$ are unknown stationary distributions.
- At each time step $t = 1, \ldots, T$, the agent,
  - chooses an arm $i(t)$, and
  - receives a numerical reward $r(t) \sim X_{i(t)}$.

Assumptions in our bandit model so far...

- Reward distributions are stationary.
- Rewards are generated by a stochastic process.

Assumptions in our Bandits Model so far

Stationary Stochastic Bandits

- Number of arms $= K$
- Reward for arm $a \sim X_a$ with mean μ_a.
- $X_1, X_2, \dots X_K$ are unknown stationary distributions.
- At each time step $t = 1, \dots, T$, the agent,
  - chooses an arm $i(t)$, and
  - receives a numerical reward $r(t) \sim X_{i(t)}$.

Assumptions in our bandit model so far...

- Reward distributions are stationary.
- Rewards are generated by a stochastic process.
- Learner can only select one arm at a time and sees absolute feedback.

Assumptions in our Bandits Model so far

Stationary Stochastic Bandits

- Number of arms $= K$
- Reward for arm $a \sim X_a$ with mean μ_a.
- $X_1, X_2, \ldots X_K$ are unknown stationary distributions.
- At each time step $t = 1, \ldots, T$, the agent,
  - chooses an arm $i(t)$, and
  - receives a numerical reward $r(t) \sim X_{i(t)}$.

Assumptions in our bandit model so far...

- Reward distributions are stationary.
- Rewards are generated by a stochastic process.
- Learner can only select one arm at a time and sees absolute feedback.
- Learner has no extra information about the arms.

Lecture 4: Outline

- Non-stationary Stochastic Bandits.
- Adversarial Bandits.
- Dueling Bandits (and a Lower Bound)
- Contextual Bandits.

Non-stationary Stochastic Bandits

Non-stationary Reward Distributions

- ~~Reward distributions are stationary~~ Non-stationary Stochastic Bandits.
- Rewards are assumed to be generated by a stochastic process.
- Learner can only select one arm at a time and sees absolute feedback.
- Learner has no extra information about the arms.

Non-stationary Stochastic Rewards

- Number of arms $= K$.
- At time step t, Reward for arm $a \sim X_a(t)$ with mean $\mu_a(t)$.
- For some t's, $\mu_a(t) \neq \mu_a(t+1)$.
- How could we characterize these changes?

Characterization of Non-stationarity

- **Bound the number of changes.**
  - Mean rewards change at unknown time steps called change-points and remain constant between two change-points.
  - Number of change-points $\leq M$.

Characterization of Non-stationarity

- **Bound the number of changes.**
  - Mean rewards change at unknown time steps called change-points and remain constant between two change-points.
  - Number of change-points $\leq M$.
- **Bound the variation in mean rewards.**
  - Mean rewards can change an arbitrary number of times, but total variation is bounded i.e.,

$$\max_a \sum_{t=1}^{T-1} |\mu_a(t) - \mu_a(t+1)| \leq V.$$

Algorithm for Non-stationary Stochastic Bandits (with a bound on the number of changes)

- Algorithm needs to forget the history before the change.

Algorithm for Non-stationary Stochastic Bandits (with a bound on the number of changes)

- Algorithm needs to forget the history before the change.
- While computing empirical mean rewards, only consider the last τ time steps.

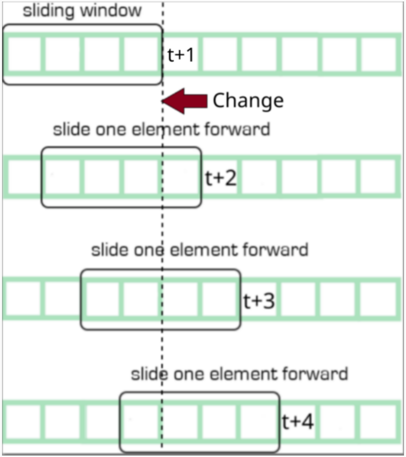
Algorithm for Non-stationary Stochastic Bandits (with a bound on the number of changes)

- Algorithm needs to forget the history before the change.
- While computing empirical mean rewards, only consider the last τ time steps.
- $\tau =$ size of the window.

Algorithm Sliding Window-UCB algorithm [Garivier and Moulines, 2011]

for $t = 1, \ldots, K$ **do**
    Choose each arm once.
end for
for $t = K + 1, \ldots$ **do**
    Compute empirical means $\hat{\mu}_1(t-1), \ldots, \hat{\mu}_K(t-1)$ based on last τ time steps.
    Select arm $i(t) = \arg\max_a \left[\hat{\mu}_a(t-1) + \text{confidence term}\right]$.
end for

How Does Sliding Window-UCB Work?



- After a change occurs, sliding window forgets the past and considers history from the current setting.

Adversarial Bandits

Adversarial Bandits

- Reward distributions are stationary.
- ~~Rewards are generated by a stochastic process~~ Adversarial bandits.
- Learner can only select one arm at a time and sees absolute feedback.
- Learner has no extra information about the arms.

Adversarial Rewards: Simple Example

A bandit game between the learner and an *adversary*.

Adversarial Rewards: Simple Example

A bandit game between the learner and an *adversary*.

Horizon $T = 1$ and number of arms $= 2$.

Adversarial Rewards: Simple Example

A bandit game between the learner and an *adversary*.

Horizon $T = 1$ and number of arms $= 2$.

Learner's goal: Minimize the learner's regret.

Adversary's goal: Maximize the learner's regret.

Adversarial Rewards : Simple Example

A bandit game between the learner and an *adversary*.

Horizon $T = 1$ and number of arms $= 2$.

Learner's goal : Minimize the learner's regret.

Adversary's goal : Maximize the learner's regret.

1. The learner tells their policy to the adversary.

Adversarial Rewards: Simple Example

A bandit game between the learner and an *adversary*.

Horizon $T = 1$ and number of arms $= 2$.

Learner's goal: Minimize the learner's regret.

Adversary's goal: Maximize the learner's regret.

1. The learner tells their policy to the adversary.
2. The learner selects an arm i according to their policy.

Adversarial Rewards: Simple Example

A bandit game between the learner and an *adversary*.

Horizon $T = 1$ and number of arms $= 2$.

Learner's goal: Minimize the learner's regret.

Adversary's goal: Maximize the learner's regret.

1. The learner tells their policy to the adversary.
2. The learner selects an arm i according to their policy.
3. The adversary observes the selected arm and secretly chooses rewards,
    - for arm 1, reward x_1 from $\{0, 1\}$, and
    - for arm 2, reward x_2 from $\{0, 1\}$.

Adversarial Rewards: Simple Example

A bandit game between the learner and an *adversary*.

Horizon $T = 1$ and number of arms $= 2$.

Learner's goal: Minimize the learner's regret.

Adversary's goal: Maximize the learner's regret.

1. The learner tells their policy to the adversary.
2. The learner selects an arm i according to their policy.
3. The adversary observes the selected arm and secretly chooses rewards,
    - for arm 1, reward x_1 from $\{0, 1\}$, and
    - for arm 2, reward x_2 from $\{0, 1\}$.
4. The learner receives reward $r = x_i$.

Adversarial Rewards: Simple Example

A bandit game between the learner and an *adversary*.

Horizon $T = 1$ and number of arms $= 2$.

Learner's goal: Minimize the learner's regret.

Adversary's goal: Maximize the learner's regret.

1. The learner tells their policy to the adversary.
2. The learner selects an arm i according to their policy.
3. The adversary observes the selected arm and secretly chooses rewards,
   - for arm 1, reward x_1 from $\{0, 1\}$, and
   - for arm 2, reward x_2 from $\{0, 1\}$.
4. The learner receives reward $r = x_i$.
5. The regret is $\mathfrak{R} = \max\{x_1, x_2\} - r$.

Adversarial Rewards: Simple Example

A bandit game between the learner and an *adversary*.

Horizon $T = 1$ and number of arms $= 2$.

Learner's goal: Minimize the learner's regret.

Adversary's goal: Maximize the learner's regret.

1. The learner tells their policy to the adversary.
2. The learner selects an arm i according to their policy.
3. The adversary observes the selected arm and secretly chooses rewards,
   - for arm 1, reward x_1 from $\{0, 1\}$, and
   - for arm 2, reward x_2 from $\{0, 1\}$.
4. The learner receives reward $r = x_i$.
5. The regret is $\mathfrak{R} = \max\{x_1, x_2\} - r$.

No matter what the learner does, the adversary can always cause linear regret for the learner.

Adversarial Rewards with Oblivious Adversary

A bandit game between the learner and the adversary.
Horizon $T = 1$ and number of arms $= 2$.

Adversarial Rewards with Oblivious Adversary

A bandit game between the learner and the adversary.
Horizon $T = 1$ and number of arms $= 2$.

1. The learner tells their policy to the the adversary.

Adversarial Rewards with Oblivious Adversary

A bandit game between the learner and the adversary.
Horizon $T = 1$ and number of arms $= 2$.

1. The learner tells their policy to the the adversary.
2. The adversary secretly chooses rewards,
   - for arm 1, reward x_1 from $\{0, 1\}$, and
   - for arm 2, reward x_2 from $\{0, 1\}$.

Adversarial Rewards with Oblivious Adversary

A bandit game between the learner and the adversary.
Horizon $T = 1$ and number of arms $= 2$.

1. The learner tells their policy to the the adversary.
2. The adversary secretly chooses rewards,
   - for arm 1, reward x_1 from $\{0, 1\}$, and
   - for arm 2, reward x_2 from $\{0, 1\}$.
3. The learner selects an arm i according to their policy.

Adversarial Rewards with Oblivious Adversary

A bandit game between the learner and the adversary.
Horizon $T = 1$ and number of arms $= 2$.

1. The learner tells their policy to the the adversary.
2. The adversary secretly chooses rewards,
   - for arm 1, reward x_1 from $\{0, 1\}$, and
   - for arm 2, reward x_2 from $\{0, 1\}$.
3. The learner selects an arm i according to their policy.
4. The learner receives reward $r = x_i$.

Adversarial Rewards with Oblivious Adversary

A bandit game between the learner and the adversary.
Horizon $T = 1$ and number of arms $= 2$.

1. The learner tells their policy to the the adversary.
2. The adversary secretly chooses rewards,
   - for arm 1, reward x_1 from $\{0, 1\}$, and
   - for arm 2, reward x_2 from $\{0, 1\}$.
3. The learner selects an arm i according to their policy.
4. The learner receives reward $r = x_i$.
5. The regret is $\mathfrak{R} = \max\{x_1, x_2\} - r$.

Adversarial Rewards with Oblivious Adversary

A bandit game between the learner and the adversary.
Horizon $T = 1$ and number of arms $= 2$.

1. The learner tells their policy to the the adversary.
2. The adversary secretly chooses rewards,
   - for arm 1, reward x_1 from $\{0, 1\}$, and
   - for arm 2, reward x_2 from $\{0, 1\}$.
3. The learner selects an arm i according to their policy.
4. The learner receives reward $r = x_i$.
5. The regret is $\mathfrak{R} = \max\{x_1, x_2\} - r$.

What happens if the the learner's policy is deterministic?

Oblivious Adversarial Rewards: Deterministic Policy

Adversarial Rewards with Oblivious Adversary

1. The learner tells their policy to the the adversary.
2. The adversary secretly chooses rewards,
   - for arm 1, reward x_1 from $\{0, 1\}$, and
   - for arm 2, reward x_2 from $\{0, 1\}$.
3. The learner selects an arm i according to their policy.
4. The learner receives reward $r = x_i$.
5. The regret is $\mathfrak{R} = \max\{x_1, x_2\} - r$.

- If the learner implements a deterministic policy e.g., play arm 1,

  the adversary can choose $x_1 = 0$ and $x_2 = 1$, since the adversary knows the learner's policy and,

  the learner's regret is 1.

Oblivious Adversarial Rewards: Deterministic Policy

Adversarial Rewards with Oblivious Adversary

1. The learner tells their policy to the the adversary.
2. The adversary secretly chooses rewards,
   - for arm 1, reward x_1 from $\{0, 1\}$, and
   - for arm 2, reward x_2 from $\{0, 1\}$.
3. The learner selects an arm i according to their policy.
4. The learner receives reward $r = x_i$.
5. The regret is $\mathfrak{R} = \max\{x_1, x_2\} - r$.

- If the learner implements a deterministic policy e.g., play arm 1,

  the adversary can choose $x_1 = 0$ and $x_2 = 1$, since the adversary knows the learner's policy and,

  the learner's regret is 1.

- Deterministic policies cause linear regret!

Oblivious Adversarial Rewards: Randomized Policy

Adversarial Rewards with Oblivious Adversary

1. The learner tells their policy to the the adversary.
2. The adversary secretly chooses rewards,
   - for arm 1, reward x_1 from $\{0, 1\}$, and
   - for arm 2, reward x_2 from $\{0, 1\}$.
3. The learner selects an arm i according to their policy.
4. The learner receives reward $r = x_i$.
5. The regret is $\mathfrak{R} = \max\{x_1, x_2\} - r$.

- If the learner implements a randomized policy (e.g., play arm 1 with probability 0.5),

  the best the adversary can do is set $x_1 = 1$ and $x_2 = 0$, and

  the learner's expected regret $= \max\{x_1, x_2\} - \mathbb{E}[r] = 1/2$.

Oblivious Adversarial Rewards: Randomized Policy

Adversarial Rewards with Oblivious Adversary

1. The learner tells their policy to the the adversary.
2. The adversary secretly chooses rewards,
   - for arm 1, reward x_1 from $\{0, 1\}$, and
   - for arm 2, reward x_2 from $\{0, 1\}$.
3. The learner selects an arm i according to their policy.
4. The learner receives reward $r = x_i$.
5. The regret is $\mathfrak{R} = \max\{x_1, x_2\} - r$.

- If the learner implements a randomized policy (e.g., play arm 1 with probability 0.5),

  the best the adversary can do is set $x_1 = 1$ and $x_2 = 0$, and

  the learner's expected regret $= \max\{x_1, x_2\} - \mathbb{E}[r] = 1/2$.
- Randomized policies can achieve sub-linear regret.☺

Adversarial Bandits

- Number of arms = K and time horizon T.

Adversarial Bandits

- Number of arms $= K$ and time horizon T.
- The adversary/environment chooses a sequence of reward vectors $\mathbf{x}(t) = (x_1(t), \ldots, x_K(t))$ for $t = 1, \ldots, T$.

Adversarial Bandits

- Number of arms $= K$ and time horizon T.
- The adversary/environment chooses a sequence of reward vectors $\mathbf{x}(t) = (x_1(t), \ldots, x_K(t))$ for $t = 1, \ldots, T$.
- At time steps $t = 1, \ldots, T$,
    - the learner selects an arm $i(t)$;

Adversarial Bandits

- Number of arms = K and time horizon T.
- The adversary/environment chooses a sequence of reward vectors $\mathbf{x}(t) = (x_1(t), \ldots, x_K(t))$ for $t = 1, \ldots, T$.
- At time steps $t = 1, \ldots, T$,
  - the learner selects an arm $i(t)$;
  - the learner receives reward $r(t) := x_{i(t)}(t)$.

Adversarial Bandits

- Number of arms $= K$ and time horizon T.
- The adversary/environment chooses a sequence of reward vectors $\mathbf{x}(t) = (x_1(t), \ldots, x_K(t))$ for $t = 1, \ldots, T$.
- At time steps $t = 1, \ldots, T$,
    - the learner selects an arm $i(t)$;
    - the learner receives reward $r(t) := x_{i(t)}(t)$.
- Performance measure?

Performance Measure : Regret I

- Recall that for stationary stochastic bandits, the goal was to minimize

$$\mathfrak{R}_\pi(T) := \underbrace{T\mu^*}_{\text{Optimal expected cumulative reward}} - \underbrace{\mathbb{E}\left[\sum_{t=1}^{T} r(t) \mid \pi\right]}_{\text{Expected cumulative reward of } \pi} .$$

Performance Measure : Regret I

- Recall that for stationary stochastic bandits, the goal was to minimize

$$\mathfrak{R}_\pi(T) := \underbrace{T\mu^*}_{\text{Optimal expected cumulative reward}} - \underbrace{\mathbb{E}\left[\sum_{t=1}^{T} r(t) \mid \pi\right]}_{\text{Expected cumulative reward of } \pi} .$$

- Benchmark policy : 'always play the arm with highest mean reward'.

Performance Measure : Regret I

- Recall that for stationary stochastic bandits, the goal was to minimize

$$\mathfrak{R}_\pi(T) := \underbrace{T\mu^*}_{\text{Optimal expected cumulative reward}} - \underbrace{\mathbb{E}\left[\sum_{t=1}^{T} r(t) \mid \pi\right]}_{\text{Expected cumulative reward of } \pi}.$$

- Benchmark policy : 'always play the arm with highest mean reward'.
- Does that make any sense for adversarial rewards?

Performance Measure : Regret I

- Recall that for stationary stochastic bandits, the goal was to minimize

$$\mathfrak{R}_\pi(T) := \underbrace{T\mu^*}_{\text{Optimal expected cumulative reward}} - \underbrace{\mathbb{E}\left[\sum_{t=1}^{T} r(t) \mid \pi\right]}_{\text{Expected cumulative reward of } \pi}.$$

- Benchmark policy : 'always play the arm with highest mean reward'.
- Does that make any sense for adversarial rewards? 

Performance Measure : Regret I

- Recall that for stationary stochastic bandits, the goal was to minimize

$$\mathfrak{R}_\pi(T) := \underbrace{T\mu^*}_{\text{Optimal expected cumulative reward}} - \underbrace{\mathbb{E}\left[\sum_{t=1}^{T} r(t) \mid \pi\right]}_{\text{Expected cumulative reward of } \pi}.$$

- Benchmark policy : ‘always play the arm with highest mean reward’.
- Does that make any sense for adversarial rewards? 

- Competing with the policy that always plays the best arm?

Performance Measure : Regret II

- The benchmark policy : 'always play the best arm (in hindsight)',

$$\text{the best arm} = \arg\max_a \sum_{t=1}^{T} x_a(t).$$

Performance Measure : Regret II

- The benchmark policy : 'always play the best arm (in hindsight)',

$$\text{the best arm} = \arg\max_a \sum_{t=1}^{T} x_a(t).$$

- The cumulative reward of the benchmark policy $= \max_a \sum_{t=1}^{T} x_a(t)$.

Performance Measure : Regret II

- The benchmark policy : 'always play the best arm (in hindsight)',

$$\text{the best arm} = \arg\max_a \sum_{t=1}^{T} x_a(t).$$

- The cumulative reward of the benchmark policy $= \max_a \sum_{t=1}^{T} x_a(t)$.
- The learner's goal is to minimize the expected cumulative regret.

$$\mathfrak{R}_\pi(T) := \max_a \sum_{t=1}^{T} x_a(t) - \mathop{\mathbb{E}}_{\pi}\left[\sum_{t=1}^{T} r(t) \mid \pi\right]$$

Algorithm for Adversarial Bandits : EXP3

- Assigns weight to each arm.
- Higher weight $\implies$ higher selection probability.
- γ = exploration parameter.
- Weight of the selected arm is updated via an estimator.
- Arms producing more rewards receive higher weights.

Algorithm EXP3 [Auer et al., 2003]

For each arm a, initialize $w_a(1) = 1$.

At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

Pick $i(t) \sim \mathbf{p(t)} = (p_1, \ldots, p_K)$.
Receive reward $r(t) := x_{i(t)}(t)$.
for $a = 1, 2, \ldots, K$ **do**

$$\hat{x}_a(t) = \begin{cases} \frac{r(t)}{p_a(t)} & \text{if } a = i(t) \\ 0 & \text{otherwise} \end{cases}$$

$$w_a(t+1) \leftarrow w_a(t) \exp\left(\frac{\gamma}{K}\hat{x}_a(t)\right).$$

end for

Algorithm for Adversarial Bandits: EXP3

- Assigns weight to each arm.
- Higher weight $\implies$ higher selection probability.
- $\gamma =$ exploration parameter.
- Weight of the selected arm is updated via an estimator.
- Arms producing more rewards receive higher weights.

Algorithm EXP3 [Auer et al., 2003]

For each arm a, initialize $w_a(1) = 1$.

At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

Pick $i(t) \sim \mathbf{p}(\mathbf{t}) = (p_1, \ldots, p_K)$.

Receive reward $r(t) := x_{i(t)}(t)$.

for $a = 1, 2, \ldots, K$ **do**

$$\hat{x}_a(t) = \begin{cases} \frac{r(t)}{p_a(t)} & \text{if } a = i(t) \\ 0 & \text{otherwise} \end{cases}$$

$w_a(t+1) \leftarrow w_a(t) \exp\left(\frac{\gamma}{K}\hat{x}_a(t)\right)$.

end for

Algorithm for Adversarial Bandits: EXP3

- Assigns weight to each arm.
- Higher weight $\implies$ higher selection probability.
- $\gamma =$ exploration parameter.
- Weight of the selected arm is updated via an estimator.
- Arms producing more rewards receive higher weights.

Algorithm EXP3 [Auer et al., 2003]

For each arm a, initialize $w_a(1) = 1$.

At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

Pick $i(t) \sim \mathbf{p(t)} = (p_1, \ldots, p_K)$.

Receive reward $r(t) := x_{i(t)}(t)$.

for $a = 1, 2, \ldots, K$ **do**

$$\hat{x}_a(t) = \begin{cases} \frac{r(t)}{p_a(t)} & \text{if } a = i(t) \\ 0 & \text{otherwise} \end{cases}$$

$w_a(t+1) \leftarrow w_a(t) \exp\left(\frac{\gamma}{K}\hat{x}_a(t)\right)$.

end for

Algorithm for Adversarial Bandits: EXP3

- Assigns weight to each arm.
- Higher weight $\implies$ higher selection probability.
- γ = exploration parameter.
- Weight of the selected arm is updated via an estimator.
- Arms producing more rewards receive higher weights.

Algorithm EXP3 [Auer et al., 2003]

1: For each arm a, initialize $w_a(1) = 1$.

2: At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

3: Pick $i(t) \sim \mathbf{p(t)} = (p_1, \ldots, p_K)$.

4: Receive reward $r(t) := x_{i(t)}(t)$.

5: **for** $a = 1, 2, \ldots, K$ **do**

6: $\hat{x}_a(t) = \begin{cases} \frac{r(t)}{p_a(t)} & \text{if } a = i(t) \\ 0 & \text{otherwise} \end{cases}$

7: $w_a(t+1) \leftarrow w_a(t) \exp\left(\frac{\gamma}{K}\hat{x}_a(t)\right)$.

8: **end for**

Algorithm for Adversarial Bandits: EXP3

- Assigns weight to each arm.
- Higher weight $\implies$ higher selection probability.
- $\gamma =$ exploration parameter.
- Weight of the selected arm is updated via an estimator.
- Arms producing more rewards receive higher weights.

Algorithm EXP3 [Auer et al., 2003]

For each arm a, initialize $w_a(1) = 1$.

At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

Pick $i(t) \sim \mathbf{p(t)} = (p_1, \dots, p_K)$.

Receive reward $r(t) \coloneqq x_{i(t)}(t)$.

for $a = 1, 2, \dots, K$ **do**

$$\hat{x}_a(t) = \begin{cases} \frac{r(t)}{p_a(t)} & \text{if } a = i(t) \\ 0 & \text{otherwise} \end{cases}$$

$$w_a(t+1) \leftarrow w_a(t) \exp\left(\tfrac{\gamma}{K}\hat{x}_a(t)\right).$$

end for

Regret Bound for EXP3

Theorem (Auer et al. [2003])

The expected cumulative regret of EXP3 is $\mathcal{O}\left(\sqrt{TK\log(K)}\right)$.

Key Lemma in the Regret Analysis

Let "history"

$$\mathcal{F}_t := i(1), i(2), \ldots, i(t-1).$$

Algorithm EXP3 Auer et al. [2003]

For each arm a, initialize $w_a(1) = 1$.
At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

Pick $i(t) \sim \mathbf{p}(\mathbf{t}) = (p_1, \ldots, p_K)$.
Receive reward $r(t) := x_{i(t)}(t)$.
for $a = 1, 2, \ldots, K$ **do**

$$\hat{x}_a(t) = \begin{cases} \frac{r(t)}{p_a(t)} & \text{if } a = i(t) \\ 0 & \text{otherwise} \end{cases}$$

$w_a(t+1) \leftarrow w_a(t) \cdot \exp \frac{\gamma}{K} \hat{x}_a(t)$.
end for

Key Lemma in the Regret Analysis

Let "history"

$\mathcal{F}_t := i(1), i(2), \ldots, i(t-1).$

Lemma

$\mathbb{E}[\hat{x}_a(t) \mid \mathcal{F}_t] = x_a(t).$

- $\hat{x}_a(t)$ estimates the reward of arm a at time t.

Algorithm EXP3 Auer et al. [2003]

1: For each arm a, initialize $w_a(1) = 1$.

2: At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

3: Pick $i(t) \sim \mathbf{p(t)} = (p_1, \ldots, p_K)$.

4: Receive reward $r(t) := x_{i(t)}(t)$.

5: **for** $a = 1, 2, \ldots, K$ **do**

6: $\hat{x}_a(t) = \begin{cases} \frac{r(t)}{p_a(t)} & \text{if } a = i(t) \\ 0 & \text{otherwise} \end{cases}$

7: $w_a(t+1) \leftarrow w_a(t) \cdot \exp \frac{\gamma}{K} \hat{x}_a(t)$.

8: **end for**

Key Lemma in the Regret Analysis

Let "history"

$\mathcal{F}_t := i(1), i(2), \ldots, i(t-1).$

Lemma

$\mathbb{E}[\hat{x}_a(t) \mid \mathcal{F}_t] = x_a(t).$

- $\hat{x}_a(t)$ estimates the reward of arm a at time t.

Proof.

$$\mathbb{E}[\hat{x}_a(t)]$$
$$= \left[p_a(t) \cdot \frac{x_a(t)}{p_a(t)} + (1 - p_a(t)) \cdot 0\right]$$
$$= x_a(t)$$

□

Algorithm EXP3 Auer et al. [2003]

1: For each arm a, initialize $w_a(1) = 1$.
2: At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

3: Pick $i(t) \sim \mathbf{p}(\mathbf{t}) = (p_1, \ldots, p_K)$.
4: Receive reward $r(t) := x_{i(t)}(t)$.
5: **for** $a = 1, 2, \ldots, K$ **do**
6: $\quad \hat{x}_a(t) = \begin{cases} \frac{r(t)}{p_a(t)} & \text{if } a = i(t) \\ 0 & \text{otherwise} \end{cases}$
7: $\quad w_a(t+1) \leftarrow w_a(t) \cdot \exp \frac{\gamma}{K} \hat{x}_a(t)$.
8: **end for**

Break

We start again after a break.

Dueling Bandits

Dueling Bandits

- Reward distributions are stationary.
- Rewards are assumed to be generated by a stochastic process.
- ~~Learner can only select one arm at a time and sees absolute feedback~~ Dueling bandits.
- Learner has no extra information about the arms.

Feedback to the Learner?

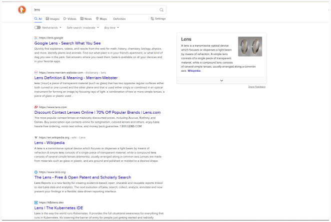


Figure 1: DuckDuckGo search results

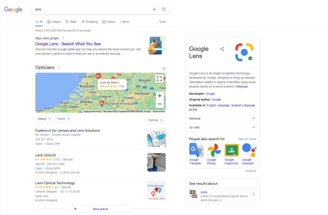


Figure 2: Google search results

- So far, we have assumed the feedback is absolute.

Feedback to the Learner?

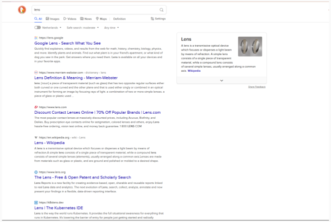


Figure 1: DuckDuckGo search results

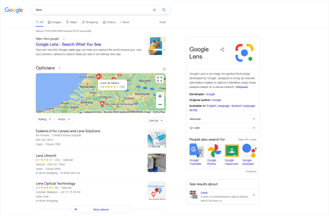


Figure 2: Google search results

- So far, we have assumed the feedback is absolute.
- What if feedback is relative and not absolute?

Feedback to the Learner?

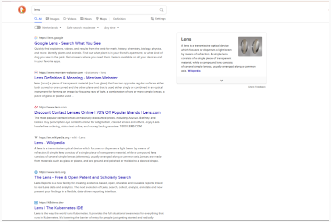


Figure 1: DuckDuckGo search results

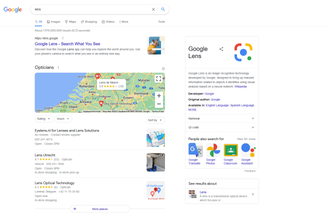


Figure 2: Google search results

- So far, we have assumed the feedback is absolute.
- What if feedback is relative and not absolute?
- Practical scenario : Information retrieval in search engines.

Feedback to the Learner?

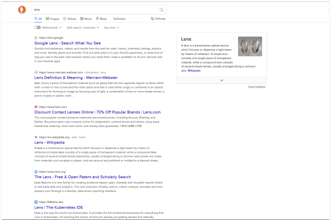


Figure 1: DuckDuckGo search results

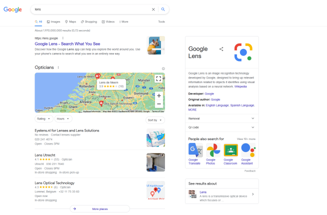


Figure 2: Google search results

- So far, we have assumed the feedback is absolute.
- What if feedback is relative and not absolute?
- Practical scenario : Information retrieval in search engines.
- Relative feedback by interleaved filtering [Radlinski and Joachims, 2007]

Adversarial Dueling Bandits: Mathematical Setting

- Number of arms $= K$ and time horizon T.

Adversarial Dueling Bandits: Mathematical Setting

- Number of arms $= K$ and time horizon T.
- The adversary/environment chooses a sequence of reward vectors $\mathbf{x}(t) = (x_1(t), \ldots, x_K(t))$ for $t = 1, \ldots, T$.

Adversarial Dueling Bandits: Mathematical Setting

- Number of arms $= K$ and time horizon T.
- The adversary/environment chooses a sequence of reward vectors $\mathbf{x}(t) = (x_1(t), \ldots, x_K(t))$ for $t = 1, \ldots, T$.
- At time steps $t = 1, \ldots, T$,
  - the learner selects two arms $i(t)$ and $j(t)$;

Adversarial Dueling Bandits : Mathematical Setting

- Number of arms $= K$ and time horizon T.
- The adversary/environment chooses a sequence of reward vectors $\mathbf{x}(t) = (x_1(t), \ldots, x_K(t))$ for $t = 1, \ldots, T$.
- At time steps $t = 1, \ldots, T$,
  - the learner selects two arms $i(t)$ and $j(t)$;
  - the learner receives (hidden) reward $r(t) := \frac{x_{i(t)}(t) + x_{j(t)}(t)}{2}$; and

Adversarial Dueling Bandits: Mathematical Setting

- Number of arms $= K$ and time horizon T.
- The adversary/environment chooses a sequence of reward vectors $\mathbf{x}(t) = (x_1(t), \ldots, x_K(t))$ for $t = 1, \ldots, T$.
- At time steps $t = 1, \ldots, T$,
  - the learner selects two arms $i(t)$ and $j(t)$;
  - the learner receives (hidden) reward $r(t) := \frac{x_{i(t)}(t) + x_{j(t)}(t)}{2}$; and
  - the learner sees relative feedback $f(t) := \psi(x_{i(t)} - x_{j(t)})$ where ψ is some feedback function.

Adversarial Dueling Bandits: Mathematical Setting

- Number of arms $= K$ and time horizon T.
- The adversary/environment chooses a sequence of reward vectors $\mathbf{x}(t) = (x_1(t), \ldots, x_K(t))$ for $t = 1, \ldots, T$.
- At time steps $t = 1, \ldots, T$,
  - the learner selects two arms $i(t)$ and $j(t)$;
  - the learner receives (hidden) reward $r(t) := \frac{x_{i(t)}(t) + x_{j(t)}(t)}{2}$; and
  - the learner sees relative feedback $f(t) := \psi(x_{i(t)} - x_{j(t)})$ where ψ is some feedback function.
- Performance measure?

Performance Measure : Regret

- The benchmark policy : 'always play the best arm (in hindsight)'.

Performance Measure : Regret

- The benchmark policy : 'always play the best arm (in hindsight)'.
- The cumulative reward of the benchmark policy $= \max_a \sum_{t=1}^{T} x_a(t)$,

Performance Measure: Regret

- The benchmark policy: 'always play the best arm (in hindsight)'.
- The cumulative reward of the benchmark policy $= \max_a \sum_{t=1}^{T} x_a(t)$,
- The learner's goal is to minimize the expected cumulative regret.

$$\mathfrak{R}_\pi(T) := \max_a \sum_{t=1}^{T} x_a(t) - \mathop{\mathbb{E}}_{\pi}\left[\sum_{t=1}^{T} r(t) \mid \pi\right]$$

Performance Measure : Regret

- The benchmark policy : ‘always play the best arm (in hindsight)’.
- The cumulative reward of the benchmark policy $= \max_a \sum_{t=1}^{T} x_a(t)$,
- The learner’s goal is to minimize the expected cumulative regret.

$$\mathfrak{R}_\pi(T) := \max_a \sum_{t=1}^{T} x_a(t) - \mathop{\mathbb{E}}_{\pi}\left[\sum_{t=1}^{T} r(t) \mid \pi\right]$$
$$= \max_a \sum_{t=1}^{T} x_a(t) - \mathop{\mathbb{E}}_{\pi}\left[\sum_{t=1}^{T} \frac{x_{i(t)}(t) + x_{j(t)}(t)}{2}\right]$$

(where $i(t)$ and $j(t)$ are the arms picked at time t).

Lower Bound

- Lower bound of a problem shows the best performance any algorithm can achieve for that problem.

Lower Bound

- Lower bound of a problem shows the best performance any algorithm can achieve for that problem.
- Lower bound tells you the *hardness* of the problem.

Lower Bound

- Lower bound of a problem shows the best performance any algorithm can achieve for that problem.
- Lower bound tells you the *hardness* of the problem.
- If upper bound of an algorithm $\simeq$ lower bound, then, the algorithm is (close) to optimal.

Lower Bound

- Lower bound of a problem shows the best performance any algorithm can achieve for that problem.
- Lower bound tells you the *hardness* of the problem.
- If upper bound of an algorithm $\simeq$ lower bound, then, the algorithm is (close) to optimal.
- Form of a typical lower bound : *For any algorithm $\mathcal{A}$, there exists an instance of the problem such that regret of $\mathcal{A}$ is at least . . . .*

Lower Bound

- Lower bound of a problem shows the best performance any algorithm can achieve for that problem.
- Lower bound tells you the *hardness* of the problem.
- If upper bound of an algorithm $\simeq$ lower bound, then, the algorithm is (close) to optimal.
- Form of a typical lower bound : *For any algorithm $\mathcal{A}$, there exists an instance of the problem such that regret of $\mathcal{A}$ is at least . . . .*
- Lower bound for stationary stochastic bandits $= \sqrt{KT}$ [Auer et al., 2003]

Reducing Problem A to Problem B

- Problem A is *reducible* to problem B,
  if an algorithm for solving problem B efficiently could also be used as a subroutine to solve problem A efficiently.

Reducing Problem A to Problem B

- Problem A is *reducible* to problem B, if an algorithm for solving problem B efficiently could also be used as a subroutine to solve problem A efficiently.
- When this is true, solving A *cannot be harder* than solving B; i.e., solving B is *at least as hard as* solving A.

For more information, click here.

Reducing Problem A to Problem B

- Problem A is *reducible* to problem B,
  if an algorithm for solving problem B efficiently could also be used as a subroutine to solve problem A efficiently.
- When this is true, solving A *cannot be harder* than solving B;
  i.e., solving B is *at least as hard as* solving A.

  For more information, click here.
- Idea : Reduce stationary stochastic bandits to dueling bandits.

Reducing Problem A to Problem B

- Problem A is *reducible* to problem B,
  if an algorithm for solving problem B efficiently could also be used as a subroutine to solve problem A efficiently.
- When this is true, solving A *cannot be harder* than solving B;
  i.e., solving B is *at least as hard as* solving A.

  For more information, click here.
- Idea : Reduce stationary stochastic bandits to dueling bandits.
- Reduction shows that solving dueling bandits is *at least as hard as* solving stationary stochastic bandits.

Reducing Problem A to Problem B

- Problem A is *reducible* to problem B,
  if an algorithm for solving problem B efficiently could also be used as a subroutine to solve problem A efficiently.
- When this is true, solving A *cannot be harder* than solving B;
  i.e., solving B is *at least as hard as* solving A.

  For more information, click here.
- Idea : Reduce stationary stochastic bandits to dueling bandits.
- Reduction shows that solving dueling bandits is *at least as hard as* solving stationary stochastic bandits.
- Lower bound for dueling bandits = Lower bound for stationary stochastic bandits.

Lower Bound for Dueling Bandits

- A generic dueling bandits algorithm DBA with following procedures: `decide`() and `update`().

Lower Bound for Dueling Bandits

- A generic dueling bandits algorithm DBA with following procedures: `decide`() and `update`().
- A stationary stochastic bandit environment CBE with `get_reward()` procedure.

Lower Bound for Dueling Bandits

- A generic dueling bandits algorithm DBA with following procedures: `decide()` and `update()`.
- A stationary stochastic bandit environment CBE with `get_reward()` procedure.

Algorithm Reduction from stationary stochastic bandits

```
Repeat
1: (i, j) ← DBA.decide(t).
2: x_i(t) ← CBE.get_reward(i).
3: x_j(t + 1) ← CBE.get_reward(j).
4: DBA.update(t, (i, j), ψ(x_i − x_j)).
5: t = t + 2.
Until t ≥ T
```

Lower Bound for Dueling Bandits

- A generic dueling bandits algorithm DBA with following procedures: `decide()` and `update()`.
- A stationary stochastic bandit environment CBE with `get_reward()` procedure.

Algorithm Reduction from stationary stochastic bandits

Repeat

$(i, j) \leftarrow$ DBA.decide(t).

$x_i(t) \leftarrow$ CBE.get_reward(i).

$x_j(t+1) \leftarrow$ CBE.get_reward(j).

DBA.update$(t, (i, j), \psi(x_i - x_j))$.

$t = t + 2$.

Until $t \geq T$

- Cumulative reward of DBA $= \mathbb{E}\left[\sum_t \frac{x_i(t)+x_j(t+1)}{2}\right]$

Lower Bound for Dueling Bandits

- A generic dueling bandits algorithm DBA with following procedures: `decide()` and `update()`.
- A stationary stochastic bandit environment CBE with `get_reward()` procedure.

Algorithm Reduction from stationary stochastic bandits

Repeat

1: $(i, j) \leftarrow$ DBA.`decide`(t).
2: $x_i(t) \leftarrow$ CBE.`get_reward`(i).
3: $x_j(t+1) \leftarrow$ CBE.`get_reward`(j).
4: DBA.`update`$(t, (i, j), \psi(x_i - x_j))$.
5: $t = t + 2$.

Until $t \geq T$

- Cumulative reward of DBA $= \mathbb{E}\left[\sum_t \frac{x_i(t)+x_j(t+1)}{2}\right]$
- $\mathbb{E}$ [CB cumulative reward of above procedure]
  $= \mathbb{E}[\sum_t x_i(t) + x_j(t+1)] = 2 * \mathbb{E}$ [cumulative reward of DBA]

Lower Bound for Dueling Bandits

- A generic dueling bandits algorithm DBA with following procedures: `decide()` and `update()`.
- A stationary stochastic bandit environment CBE with `get_reward()` procedure.

Algorithm Reduction from stationary stochastic bandits

Repeat
$(i, j) \leftarrow$ DBA.`decide`(t).
$x_i(t) \leftarrow$ CBE.`get_reward`(i).
$x_j(t+1) \leftarrow$ CBE.`get_reward`(j).
DBA.`update`$(t, (i, j), \psi(x_i - x_j))$.
$t = t + 2$.
Until $t \geq T$

- Cumulative reward of DBA $= \mathbb{E}\left[\sum_t \frac{x_i(t)+x_j(t+1)}{2}\right]$
- $\mathbb{E}$ [CB cumulative reward of above procedure] $= \mathbb{E}[\sum_t x_i(t) + x_j(t+1)] = 2 * \mathbb{E}$ [cumulative reward of DBA]
- $\mathbb{E}$[Regret of DBA] is of the same order as $\mathbb{E}$[Regret of CB].

Algorithm for Adversarial Dueling Bandits: REX3

- Assigns weight to each arm.
- Higher weight $\implies$ higher selection probability.
- $\gamma \in (0, 0.5)$ exploration parameter.
- Weights of the selected arms are updated.
- Arms winning more duels receive higher weights.

Algorithm REX3 (PG et al.)

1: For each arm a, initialize weights

$$w_a(1) = 1.$$

2: At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

3: Sample

$$i, j \sim \mathbf{p}(\mathbf{t}) = (p_1, \ldots, p_K),$$

get $f(t) = \psi(x_i - x_j)$.

4: $w_i(t+1) \leftarrow w_i(t) \cdot e^{\frac{\gamma}{K}\frac{f(t)}{2p_i}}$

$w_j(t+1) \leftarrow w_j(t) \cdot e^{-\frac{\gamma}{K}\frac{f(t)}{2p_j}}$

Algorithm for Adversarial Dueling Bandits: REX3

- Assigns weight to each arm.
- Higher weight $\implies$ higher selection probability.
- $\gamma \in (0, 0.5)$ exploration parameter.
- Weights of the selected arms are updated.
- Arms winning more duels receive higher weights.

Algorithm REX3 (PG et al.)

1: For each arm a, initialize weights

$$w_a(1) = 1.$$

2: At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

3: Sample

$$i, j \sim \mathbf{p(t)} = (p_1, \ldots, p_K),$$

get $f(t) = \psi(x_i - x_j)$.

4: $w_i(t+1) \leftarrow w_i(t) \cdot e^{\frac{\gamma}{K}\frac{f(t)}{2p_i}}$

$w_j(t+1) \leftarrow w_j(t) \cdot e^{-\frac{\gamma}{K}\frac{f(t)}{2p_j}}$

Algorithm for Adversarial Dueling Bandits: REX3

- Assigns weight to each arm.
- Higher weight $\implies$ higher selection probability.
- $\gamma \in (0, 0.5)$ exploration parameter.
- Weights of the selected arms are updated.
- Arms winning more duels receive higher weights.

Algorithm REX3 (PG et al.)

1: For each arm a, initialize weights

$$w_a(1) = 1.$$

2: At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

3: Sample

$$i, j \sim \mathbf{p(t)} = (p_1, \ldots, p_K),$$

get $f(t) = \psi(x_i - x_j)$.

4: $w_i(t+1) \leftarrow w_i(t) \cdot e^{\frac{\gamma}{K}\frac{f(t)}{2p_i}}$

$w_j(t+1) \leftarrow w_j(t) \cdot e^{-\frac{\gamma}{K}\frac{f(t)}{2p_j}}$

Algorithm for Adversarial Dueling Bandits : REX3

- Assigns weight to each arm.
- Higher weight $\implies$ higher selection probability.
- $\gamma \in (0, 0.5)$ exploration parameter.
- Weights of the selected arms are updated.
- Arms winning more duels receive higher weights.

Algorithm REX3 (PG et al.)

1: For each arm a, initialize weights

$$w_a(1) = 1.$$

2: At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

3: Sample

$$i, j \sim \mathbf{p(t)} = (p_1, \ldots, p_K),$$

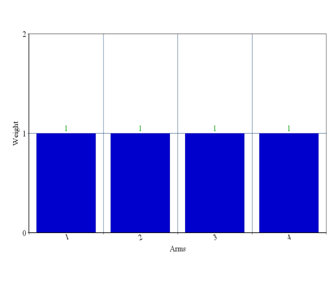
get $f(t) = \psi(x_i - x_j)$.

4: $w_i(t+1) \leftarrow w_i(t) \cdot e^{\frac{\gamma}{K}\frac{f(t)}{2p_i}}$

$w_j(t+1) \leftarrow w_j(t) \cdot e^{-\frac{\gamma}{K}\frac{f(t)}{2p_j}}$

How Does REX3 Work?

Weights at $t = 0$
($\gamma = 0.4$)



- Update weight according to (relative) feedback.

Algorithm REX3

1: For each arm a, initialize weights

$$w_a(1) = 1.$$

2: At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

3: Sample

$$i, j \sim \mathbf{p}(\mathbf{t}) = (p_1, \ldots, p_K),$$

get $f(t) = \psi(x_i - x_j)$.

4: $w_i(t+1) \leftarrow w_i(t) \cdot e^{\frac{\gamma}{K}\frac{f(t)}{2p_i}}$

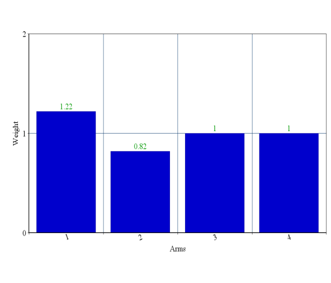
$w_j(t+1) \leftarrow w_j(t) \cdot e^{-\frac{\gamma}{K}\frac{f(t)}{2p_j}}$

How Does REX3 Work?

$i = 1$, $j = 2$

1 wins the duel

Weights at $t = 1$



- Weight may decrease.

Algorithm REXЗ

1: For each arm a, initialize weights

$$w_a(1) = 1.$$

2: At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

3: Sample

$$i, j \sim \mathbf{p}(\mathbf{t}) = (p_1, \ldots, p_K),$$

get $f(t) = \psi(x_i - x_j)$.

4: $w_i(t+1) \leftarrow w_i(t) \cdot e^{\frac{\gamma}{K}\frac{f(t)}{2p_i}}$

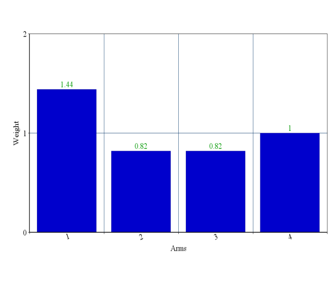
$w_j(t+1) \leftarrow w_j(t) \cdot e^{-\frac{\gamma}{K}\frac{f(t)}{2p_j}}$

How Does REX3 Work?

$i = 1, j = 3$

1 wins the duel

Weights at $t = 2$



- Weights increase at arms which win regularly.

Algorithm REX3

1: For each arm a, initialize weights

$$w_a(1) = 1.$$

2: At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

3: Sample

$$i, j \sim \mathbf{p(t)} = (p_1, \ldots, p_K),$$

get $f(t) = \psi(x_i - x_j)$.

4: $w_i(t+1) \leftarrow w_i(t) \cdot e^{\frac{\gamma}{K}\frac{f(t)}{2p_i}}$

$w_j(t+1) \leftarrow w_j(t) \cdot e^{-\frac{\gamma}{K}\frac{f(t)}{2p_j}}$

Regret Upper Bound for REX3

Theorem (PG et al., 2015)

The expected cumulative regret of $\mathrm{REX3}$ *is* $\mathcal{O}\left(\sqrt{TK\log(K)}\right)$.

Regret Upper Bound for REX3

Theorem (PG et al., 2015)

The expected cumulative regret of $\mathrm{REX3}$ *is* $\mathcal{O}\left(\sqrt{TK\log(K)}\right)$.

- The upper bound is only $\sqrt{\log(K)}$ away than the lower bound $\sqrt{KT}$ so $\mathrm{REX3}$ is near-optimal.

Analysis

- binary rewards i.e.,

$$x_a(t) = \text{either } 0 \text{ or } 1,$$

  for all arms a and all time steps t.

Analysis

- binary rewards i.e.,

$$x_a(t) = \text{either } 0 \text{ or } 1,$$

  for all arms a and all time steps t.
- Feedback function ψ is identity i.e., when arms i and j are selected,

$$\text{feedback} = f := \psi(x_i - x_j) = x_i - x_j$$

Analysis

- binary rewards i.e.,

$$x_a(t) = \text{either } 0 \text{ or } 1,$$

  for all arms a and all time steps t.
- Feedback function ψ is identity i.e., when arms i and j are selected,

$$\text{feedback} = f := \psi(x_i - x_j) = x_i - x_j$$

- When when arms i and j are selected,

$$\text{feedback} = f = \begin{cases} -1 & \text{if } x_i < x_j \\ 0 & \text{if } x_i = x_j \\ +1 & \text{if } x_i > x_j \end{cases}$$

Estimator for an arm

Algorithm REX3

1: For each arm a, initialize weights

$$w_a(1) = 1.$$

2: At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

3: Sample

$$i, j \sim \mathbf{p(t)} = (p_1, \ldots, p_K),$$

get $f(t) = \psi(x_i - x_j)$.

4: $w_i(t+1) \leftarrow w_i(t) \cdot e^{\frac{\gamma}{K}\frac{f(t)}{2p_i}}$

$w_j(t+1) \leftarrow w_j(t) \cdot e^{-\frac{\gamma}{K}\frac{f(t)}{2p_j}}$

Estimator for an arm

- Let

$$\hat{c}_a(t) := \mathbb{I}[a = i]\frac{(x_i - x_j)}{2p_i} + \mathbb{I}[a = j]\frac{(x_j - x_i)}{2p_j}$$

where i and j are the arms picked at time t.

Algorithm REX3

1: For each arm a, initialize weights

$$w_a(1) = 1.$$

2: At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

3: Sample

$$i, j \sim \mathbf{p(t)} = (p_1, \ldots, p_K),$$

get $f(t) = \psi(x_i - x_j)$.

4: $w_i(t+1) \leftarrow w_i(t) \cdot e^{\frac{\gamma}{K}\frac{f(t)}{2p_i}}$

$w_j(t+1) \leftarrow w_j(t) \cdot e^{-\frac{\gamma}{K}\frac{f(t)}{2p_j}}$

Estimator for an arm

- Let

$$\hat{c}_a(t) := \mathbb{I}[a = i]\frac{(x_i - x_j)}{2p_i} + \mathbb{I}[a = j]\frac{(x_j - x_i)}{2p_j}$$

where i and j are the arms picked at time t.

- Step 4 is equivalent to: for each arm a,

$$w_a(t+1) = w_a(t) \cdot e^{\frac{\gamma}{K}\hat{c}_a(t)}$$

Algorithm REX3

1: For each arm a, initialize weights

$$w_a(1) = 1.$$

2: At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

3: Sample

$$i, j \sim \mathbf{p(t)} = (p_1, \ldots, p_K),$$

get $f(t) = \psi(x_i - x_j)$.

4: $w_i(t+1) \leftarrow w_i(t) \cdot e^{\frac{\gamma}{K}\frac{f(t)}{2p_i}}$

$w_j(t+1) \leftarrow w_j(t) \cdot e^{-\frac{\gamma}{K}\frac{f(t)}{2p_j}}$

Main Lemma

- Step 4 is equivalent to :
  for each arm a,

  $$w_a(t+1) = w_a(t) \cdot e^{\frac{\gamma}{K}\hat{c}_a(t)}$$

Algorithm REX3

1: For each arm a, initialize weights

$$w_a(1) = 1.$$

2: At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

3: Sample

$$i, j \sim \mathbf{p}(\mathbf{t}) = (p_1, \ldots, p_K),$$

get $f(t) = \psi(x_i - x_j)$.

4: $w_i(t+1) \leftarrow w_i(t) \cdot e^{\frac{\gamma}{K}\frac{f(t)}{2p_i}}$

$w_j(t+1) \leftarrow w_j(t) \cdot e^{-\frac{\gamma}{K}\frac{f(t)}{2p_j}}$

Main Lemma

- Step 4 is equivalent to :
  for each arm a,

$$w_a(t+1) = w_a(t) \cdot e^{\frac{\gamma}{K}\hat{c}_a(t)}$$

Let $\mathcal{F}_t := i(1), j(1), \ldots, i(t), j(t)$.

Algorithm REX3

1: For each arm a, initialize weights

$$w_a(1) = 1.$$

2: At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

3: Sample

$$i, j \sim \mathbf{p}(\mathbf{t}) = (p_1, \ldots, p_K),$$

get $f(t) = \psi(x_i - x_j)$.

4: $w_i(t+1) \leftarrow w_i(t) \cdot e^{\frac{\gamma}{K}\frac{f(t)}{2p_i}}$

$w_j(t+1) \leftarrow w_j(t) \cdot e^{-\frac{\gamma}{K}\frac{f(t)}{2p_j}}$

Main Lemma

- Step 4 is equivalent to :
  for each arm a,

$$w_a(t+1) = w_a(t) \cdot e^{\frac{\gamma}{K}\hat{c}_a(t)}$$

Let $\mathcal{F}_t := i(1), j(1), \ldots, i(t), j(t)$.

Lemma (Lemma 1 in PG et al.)

$\mathbb{E}[\hat{c}_a(t) \mid \mathcal{F}_t] =$
$\mathbb{E}_{i \sim p(t)}[x_a(t) - x_i(t)]$.

- $\hat{c}_a(t)$ estimates the relative utility/advantage of picking arm a instead of picking an arm according to $\mathbf{p}(\mathbf{t})$.

Algorithm REX3

1: For each arm a, initialize weights

$$w_a(1) = 1.$$

2: At time t, for each arm a,

$$p_a \leftarrow (1-\gamma)\frac{w_a(t)}{\sum_{b=1}^{K} w_b(t)} + \frac{\gamma}{K}$$

3: Sample

$$i, j \sim \mathbf{p}(\mathbf{t}) = (p_1, \ldots, p_K),$$

get $f(t) = \psi(x_i - x_j)$.

4: $w_i(t+1) \leftarrow w_i(t) \cdot e^{\frac{\gamma}{K}\frac{f(t)}{2p_i}}$

$w_j(t+1) \leftarrow w_j(t) \cdot e^{-\frac{\gamma}{K}\frac{f(t)}{2p_j}}$

Proof of Main Lemma

Lemma (Lemma 1 in in PG et al.)

$\mathbb{E}[\hat{c}_a(t) \mid \mathcal{F}_t] = \mathbb{E}_{i \sim \mathbf{p(t)}}[x_a(t) - x_i(t)].$

Proof.

Proof of Main Lemma

Lemma (Lemma 1 in in PG et al.)

$\mathbb{E}[\hat{c}_a(t) \mid \mathcal{F}_t] = \mathbb{E}_{i \sim \mathbf{p(t)}}[x_a(t) - x_i(t)]$.

Proof. Recall number of arms $= K$, $\mathbf{p(t)}$ = arm selection probabilities, $\mathbb{E}[x] \coloneqq \sum_i i \cdot \mathbb{P}(x = i)$ and $\hat{c}_a(t) \coloneqq \mathbb{I}[a = i]\frac{(x_i - x_j)}{2p_i} + \mathbb{I}[a = j]\frac{(x_j - x_i)}{2p_j}$.

Proof of Main Lemma

Lemma (Lemma 1 in in PG et al.)

$\mathbb{E}[\hat{c}_a(t) \mid \mathcal{F}_t] = \mathbb{E}_{i\sim\mathbf{p(t)}}[x_a(t) - x_i(t)]$.

Proof. Recall number of arms $= K$, $\mathbf{p(t)}$ = arm selection probabilities, $\mathbb{E}[x] \coloneqq \sum_i i \cdot \mathbb{P}(x = i)$ and $\hat{c}_a(t) \coloneqq \mathbb{I}[a = i]\frac{(x_i - x_j)}{2p_i} + \mathbb{I}[a = j]\frac{(x_j - x_i)}{2p_j}$.

$$\mathop{\mathbb{E}}_{(i,j)\sim\mathbf{p(t)}}[\hat{c}_a(t)] = \sum_{m=1}^{K}\sum_{n=1}^{K} p_m p_n \left[\mathbb{I}[a = m]\frac{(x_m - x_n)}{2p_m} + \mathbb{I}[a = n]\frac{(x_n - x_m)}{2p_n}\right]$$

Proof of Main Lemma

Lemma (Lemma 1 in in PG et al.)

$\mathbb{E}[\hat{c}_a(t) \mid \mathcal{F}_t] = \mathbb{E}_{i\sim \mathbf{p(t)}}[x_a(t) - x_i(t)]$.

Proof. Recall number of arms $= K$, $\mathbf{p(t)}$ = arm selection probabilities, $\mathbb{E}[x] \coloneqq \sum_i i \cdot \mathbb{P}(x = i)$ and $\hat{c}_a(t) \coloneqq \mathbb{I}[a = i]\frac{(x_i - x_j)}{2p_i} + \mathbb{I}[a = j]\frac{(x_j - x_i)}{2p_j}$.

$$\mathop{\mathbb{E}}_{(i,j)\sim \mathbf{p(t)}}[\hat{c}_a(t)] = \sum_{m=1}^{K}\sum_{n=1}^{K} p_m p_n \left[\mathbb{I}[a = m]\frac{(x_m - x_n)}{2p_m} + \mathbb{I}[a = n]\frac{(x_n - x_m)}{2p_n}\right]$$

$$= \sum_{m=1}^{K}\sum_{n=1}^{K} \cancel{p_m} p_n \; \mathbb{I}[a = m]\frac{(x_m - x_n)}{2\cancel{p_m}}$$

$$+ \sum_{m=1}^{K}\sum_{n=1}^{K} p_m \cancel{p_n} \; \mathbb{I}[a = n]\frac{(x_n - x_m)}{2\cancel{p_n}}$$

Proof of Main Lemma

Lemma (Lemma 1 in in PG et al.)

$\mathbb{E}[\hat{c}_a(t) \mid \mathcal{F}_t] = \mathbb{E}_{i \sim \mathbf{p(t)}}[x_a(t) - x_i(t)]$.

Proof. Recall number of arms $= K$, $\mathbf{p(t)}$ = arm selection probabilities, $\mathbb{E}[x] \coloneqq \sum_i i \cdot \mathbb{P}(x = i)$ and $\hat{c}_a(t) \coloneqq \mathbb{I}[a = i]\frac{(x_i - x_j)}{2p_i} + \mathbb{I}[a = j]\frac{(x_j - x_i)}{2p_j}$.

$$\mathop{\mathbb{E}}_{(i,j)\sim\mathbf{p(t)}}[\hat{c}_a(t)] = \sum_{m=1}^{K}\sum_{n=1}^{K} p_m p_n \left[\mathbb{I}[a = m]\frac{(x_m - x_n)}{2p_m} + \mathbb{I}[a = n]\frac{(x_n - x_m)}{2p_n}\right]$$

$$= \sum_{m=1}^{K}\sum_{n=1}^{K} \cancel{p_m} p_n \, \mathbb{I}[a = m]\frac{(x_m - x_n)}{2\cancel{p_m}}$$

$$+ \sum_{m=1}^{K}\sum_{n=1}^{K} p_m \cancel{p_n} \, \mathbb{I}[a = n]\frac{(x_n - x_m)}{2\cancel{p_n}}$$

$$= \sum_{n=1}^{K} p_n \frac{(x_a - x_n)}{2} + \sum_{m=1}^{K} p_m \frac{(x_a - x_m)}{2}$$

Proof of Main Lemma

Lemma (Lemma 1 in in PG et al.)

$\mathbb{E}[\hat{c}_a(t) \mid \mathcal{F}_t] = \mathbb{E}_{i \sim \mathbf{p(t)}}[x_a(t) - x_i(t)]$.

Proof. Recall number of arms $= K$, $\mathbf{p(t)}$ = arm selection probabilities, $\mathbb{E}[x] \coloneqq \sum_i i \cdot \mathbb{P}(x = i)$ and $\hat{c}_a(t) \coloneqq \mathbb{I}[a = i]\frac{(x_i - x_j)}{2p_i} + \mathbb{I}[a = j]\frac{(x_j - x_i)}{2p_j}$.

$$\begin{aligned}
\mathop{\mathbb{E}}_{(i,j)\sim\mathbf{p(t)}}[\hat{c}_a(t)] &= \sum_{m=1}^{K}\sum_{n=1}^{K} p_m p_n \left[\mathbb{I}[a = m]\frac{(x_m - x_n)}{2p_m} + \mathbb{I}[a = n]\frac{(x_n - x_m)}{2p_n}\right] \\
&= \sum_{m=1}^{K}\sum_{n=1}^{K} \cancel{p_m} p_n \ \mathbb{I}[a = m]\frac{(x_m - x_n)}{2\cancel{p_m}} \\
&\quad + \sum_{m=1}^{K}\sum_{n=1}^{K} p_m \cancel{p_n} \ \mathbb{I}[a = n]\frac{(x_n - x_m)}{2\cancel{p_n}} \\
&= \sum_{n=1}^{K} p_n \frac{(x_a - x_n)}{2} + \sum_{m=1}^{K} p_m \frac{(x_a - x_m)}{2} \\
&= \mathop{\mathbb{E}}_{i\sim\mathbf{p(t)}}[x_a(t) - x_i(t)].
\end{aligned}$$

Contextual Bandits

Contextual Bandits

- Reward distributions are stationary.
- Rewards are assumed to be generated by a stochastic process.
- Learner can only select one arm at a time and sees absolute feedback.
- ~~Learner has no extra information about the arms.~~ Contextual bandits.

Availability of Extra Information

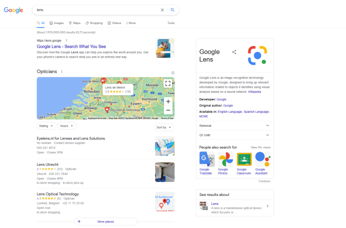


Figure 3: Google search results

- Observation of extra information (*context*) before choosing an action.
- Practical scenario : News recommendation, ad selection.

Contextual Bandits: Mathematical Setting

- At each time step $t = 1, 2, \ldots, T$

Contextual Bandits: Mathematical Setting

- At each time step $t = 1, 2, \ldots, T$
  - the learner observes feature vector (context) $\mathbf{x_t} \in \mathbb{R}^d$,

Contextual Bandits: Mathematical Setting

- At each time step $t = 1, 2, \ldots, T$
  - the learner observes feature vector (context) $\mathbf{x_t} \in \mathbb{R}^d$,
  - the learner chooses an arm $i(t)$ and,

Contextual Bandits: Mathematical Setting

- At each time step $t = 1, 2, \ldots, T$
  - the learner observes feature vector (context) $\mathbf{x_t} \in \mathbb{R}^d$,
  - the learner chooses an arm $i(t)$ and,
  - the learner receives a reward $r(t) = r_{t,i(t)}$.

Contextual Bandits: Mathematical Setting

- At each time step $t = 1, 2, \ldots, T$
  - the learner observes feature vector (context) $\mathbf{x_t} \in \mathbb{R}^d$,
  - the learner chooses an arm $i(t)$ and,
  - the learner receives a reward $r(t) = r_{t,i(t)}$.
- Linear dependence: $\mathbb{E}[r_{t,a} \mid \mathbf{x_t}] = \mathbf{x_t} \cdot \theta_a$ for some unknown vector $\theta_a \in \mathbb{R}^d$.

Example

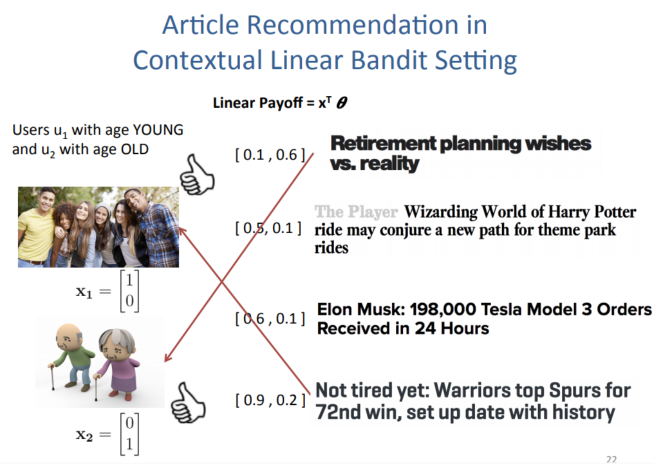


Image source: *blogpost*

Algorithm for Linear Contextual Bandits

Algorithm LinUCB [Li et al., 2010]

1: Compute confidence regions $C_{a,t}$ for each arm a.
2: Observe feature vector (context) $\mathbf{x_t} \in \mathbb{R}^d$.
3: For each arm a, compute

$$UCB_t(a \mid \mathbf{x}_{t,a}) = \sup_{\hat{\theta}_a \in C_{a,t}} \mathbf{x_t} \cdot \hat{\theta}_a$$

4: Select the arm which maximizes $UCB_t(a \mid \mathbf{x}_{t,a})$

Summary

- Non-stationary Stochastic Bandits.
- Adversarial Bandits.
- Dueling Bandits (and a Lower Bound)
- Contextual Bandits.

Next Lecture

- Reinforcement learning in Markov decision processes.
- A near-optimal algorithm : UCRL.

References

Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1): 48–77, jan 2003. ISSN 0097-5397. doi: 10.1137/S0097539701398375. URL https://doi.org/10.1137/S0097539701398375.

Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In Jyrki Kivinen, Csaba Szepesvári, Esko Ukkonen, and Thomas Zeugmann, editors, *Algorithmic Learning Theory*, pages 174–188, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-24412-4.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation, 2010.

F. Radlinski and T. Joachims. Active exploration for learning rankings from clickthrough data. In *KDD 2007*, pages 570–579. ACM, 2007. doi: 10.1145/1281192.1281254.