ECEN 743: Reinforcement Learning

Exploration in RL

Dileep Kalathil
Assistant Professor
Department of Electrical and Computer Engineering
Texas A&M University

References

- [UCB Algorithm] Chapter 7, Tor Lattimore and Csaba Szepesvári, "Bandit algorithms". Cambridge University Press, 2020.
- [Posterior Sampling Algorithm] Chapter 36, Tor Lattimore and Csaba Szepesvári, "Bandit algorithms". Cambridge University Press, 2020.

RL: Exploration vs Exploitation

- In RL, unlike supervised and unsupervised learning, data is not given a priori

RL: Exploration vs Exploitation

- In RL, unlike supervised and unsupervised learning, data is not given a priori
- Agent learns about the environment by trying things out

RL: Exploration vs Exploitation

- In RL, unlike supervised and unsupervised learning, data is not given a priori
- Agent learns about the environment by trying things out
  - RL in some way is a 'clever' trial-and-error learning

RL: Exploration vs Exploitation

- In RL, unlike supervised and unsupervised learning, data is not given a priori
- Agent learns about the environment by trying things out
  - RL in some way is a ‘clever’ trial-and-error learning
- Agent should learn a good control policy, from its experiences of the environment but without loosing too much reward along the way

RL: Exploration vs Exploitation

- In RL, unlike supervised and unsupervised learning, data is not given a priori
- Agent learns about the environment by trying things out
  - RL in some way is a ‘clever’ trial-and-error learning
- Agent should learn a good control policy, from its experiences of the environment but without loosing too much reward along the way

- Online decision-making involves a fundamental choice:

RL: Exploration vs Exploitation

- In RL, unlike supervised and unsupervised learning, data is not given a priori
- Agent learns about the environment by trying things out
  - RL in some way is a ‘clever’ trial-and-error learning
- Agent should learn a good control policy, from its experiences of the environment but without loosing too much reward along the way

- Online decision-making involves a fundamental choice:
- Exploitation: Make the best (greedy) decision given current information

RL: Exploration vs Exploitation

- In RL, unlike supervised and unsupervised learning, data is not given a priori
- Agent learns about the environment by trying things out
  - RL in some way is a 'clever' trial-and-error learning
- Agent should learn a good control policy, from its experiences of the environment but without loosing too much reward along the way

- Online decision-making involves a fundamental choice:
- Exploitation: Make the best (greedy) decision given current information
- Exploration: Gather more information to make the best decisions

RL: Exploration vs Exploitation

- In RL, unlike supervised and unsupervised learning, data is not given a priori
- Agent learns about the environment by trying things out
  - RL in some way is a ‘clever’ trial-and-error learning
- Agent should learn a good control policy, from its experiences of the environment but without loosing too much reward along the way

- Online decision-making involves a fundamental choice:
- Exploitation: Make the best (greedy) decision given current information
- Exploration: Gather more information to make the best decisions
- The optimal long-term strategy may involve sub-optimal short-term decisions

RL: Exploration vs Exploitation

- In RL, unlike supervised and unsupervised learning, data is not given a priori
- Agent learns about the environment by trying things out
  - RL in some way is a 'clever' trial-and-error learning
- Agent should learn a good control policy, from its experiences of the environment but without loosing too much reward along the way

- Online decision-making involves a fundamental choice:
- Exploitation: Make the best (greedy) decision given current information
- Exploration: Gather more information to make the best decisions
- The optimal long-term strategy may involve sub-optimal short-term decisions
- How to optimally balance exploration and exploitation?

RL: Exploration vs Exploitation

- In RL, unlike supervised and unsupervised learning, data is not given a priori
- Agent learns about the environment by trying things out
  - RL in some way is a 'clever' trial-and-error learning
- Agent should learn a good control policy, from its experiences of the environment but without loosing too much reward along the way

- Online decision-making involves a fundamental choice:
- Exploitation: Make the best (greedy) decision given current information
- Exploration: Gather more information to make the best decisions
- The optimal long-term strategy may involve sub-optimal short-term decisions
- How to optimally balance exploration and exploitation?
  - This is one of the fundamental (research) problems of RL

Exploration vs Exploitation: Examples

Exploration vs Exploitation: Examples

- Restaurant Selection

Exploration vs Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant

Exploration vs Exploitation: Examples

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

Exploration vs Exploitation: Examples

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

Exploration vs Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant
- Online advertisements
  - Exploitation: Show the most successful ad

Exploration vs Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant
- Online advertisements
  - Exploitation: Show the most successful ad
  - Exploration: Show a different ad

Exploration vs Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant
- Online advertisements
  - Exploitation: Show the most successful ad
  - Exploration: Show a different ad
- Oil Drilling

Exploration vs Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant
- Online advertisements
  - Exploitation: Show the most successful ad
  - Exploration: Show a different ad
- Oil Drilling
  - Exploitation: Drill at best known location

Exploration vs Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant
- Online advertisements
  - Exploitation: Show the most successful ad
  - Exploration: Show a different ad
- Oil Drilling
  - Exploitation: Drill at best known location
  - Exploration: Drill at new location

Exploration vs Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant
- Online advertisements
  - Exploitation: Show the most successful ad
  - Exploration: Show a different ad
- Oil Drilling
  - Exploitation: Drill at best known location
  - Exploration: Drill at new location
- Games

Exploration vs Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant
- Online advertisements
  - Exploitation: Show the most successful ad
  - Exploration: Show a different ad
- Oil Drilling
  - Exploitation: Drill at best known location
  - Exploration: Drill at new location
- Games
  - Exploitation: Play the move you believe is best

Exploration vs Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant
- Online advertisements
  - Exploitation: Show the most successful ad
  - Exploration: Show a different ad
- Oil Drilling
  - Exploitation: Drill at best known location
  - Exploration: Drill at new location
- Games
  - Exploitation: Play the move you believe is best
  - Exploration: Play an experimental move

Multi-Armed Bandits

- MAB: Canonical formulation of online (sequential) decision making problems without state dynamics that focuses on exploration vs exploitation trade-off

Multi-Armed Bandits

- MAB: Canonical formulation of online (sequential) decision making problems without state dynamics that focuses on exploration vs exploitation trade-off
  - MAB is special case of RL (with a single state)

Multi-Armed Bandits

- MAB: Canonical formulation of online (sequential) decision making problems without state dynamics that focuses on exploration vs exploitation trade-off
  - MAB is special case of RL (with a single state)

Multi-Armed Bandits

- MAB: Canonical formulation of online (sequential) decision making problems without state dynamics that focuses on exploration vs exploitation trade-off
  - MAB is special case of RL (with a single state)

- The name MAB comes from slot machines

μ_1

μ_2

Multi-Armed Bandits

- MAB: Canonical formulation of online (sequential) decision making problems without state dynamics that focuses on exploration vs exploitation trade-off
  - MAB is special case of RL (with a single state)

- The name MAB comes from slot machines
- Each machine is called an arm

μ_1 

μ_2 

Multi-Armed Bandits

- MAB: Canonical formulation of online (sequential) decision making problems without state dynamics that focuses on exploration vs exploitation trade-off
  - MAB is special case of RL (with a single state)

- The name MAB comes from slot machines
- Each machine is called an arm
- Select arm k and get a random reward with mean μ_k

μ_1

μ_2

Multi-Armed Bandits

- MAB: Canonical formulation of online (sequential) decision making problems without state dynamics that focuses on exploration vs exploitation trade-off
  - MAB is special case of RL (with a single state)

- The name MAB comes from slot machines
- Each machine is called an arm
- Select arm k and get a random reward with mean μ_k
- Only one selection at each time

μ_1 

μ_2 

Multi-Armed Bandits

- MAB: Canonical formulation of online (sequential) decision making problems without state dynamics that focuses on exploration vs exploitation trade-off
  - MAB is special case of RL (with a single state)

- The name MAB comes from slot machines
- Each machine is called an arm
- Select arm k and get a random reward with mean μ_k
- Only one selection at each time
- Only observe the outcome of that selection

μ_1 

μ_2 

Multi-Armed Bandits

- MAB: Canonical formulation of online (sequential) decision making problems without state dynamics that focuses on exploration vs exploitation trade-off
  - MAB is special case of RL (with a single state)

- The name MAB comes from slot machines
- Each machine is called an arm
- Select arm k and get a random reward with mean μ_k
- Only one selection at each time
- Only observe the outcome of that selection
- Objective: Maximize the expected cumulative reward in T successive selections

μ_1

μ_2

Multi-Armed Bandits

- MAB: Canonical formulation of online (sequential) decision making problems without state dynamics that focuses on exploration vs exploitation trade-off
  - MAB is special case of RL (with a single state)

- The name MAB comes from slot machines
- Each machine is called an arm
- Select arm k and get a random reward with mean μ_k
- Only one selection at each time
- Only observe the outcome of that selection
- Objective: Maximize the expected cumulative reward in T successive selections
- Which arm will you select?

μ_1

μ_2

Multi-Armed Bandits

- MAB: Canonical formulation of online (sequential) decision making problems without state dynamics that focuses on exploration vs exploitation trade-off
  - MAB is special case of RL (with a single state)

- The name MAB comes from slot machines
- Each machine is called an arm
- Select arm k and get a random reward with mean μ_k
- Only one selection at each time
- Only observe the outcome of that selection
- Objective: Maximize the expected cumulative reward in T successive selections
- Which arm will you select?

μ_1

μ_2

Multi-Armed Bandits

- MAB: Canonical formulation of online (sequential) decision making problems without state dynamics that focuses on exploration vs exploitation trade-off
  - MAB is special case of RL (with a single state)

- The name MAB comes from slot machines
- Each machine is called an arm
- Select arm k and get a random reward with mean μ_k
- Only one selection at each time
- Only observe the outcome of that selection
- Objective: Maximize the expected cumulative reward in T successive selections
- Which arm will you select?
- Optimal choice with complete information: select the arm with the highest mean reward

μ_1

μ_2

Multi-Armed Bandits

- MAB: Canonical formulation of online (sequential) decision making problems without state dynamics that focuses on exploration vs exploitation trade-off
  - MAB is special case of RL (with a single state)

- The name MAB comes from slot machines
- Each machine is called an arm
- Select arm k and get a random reward with mean μ_k
- Only one selection at each time
- Only observe the outcome of that selection
- Objective: Maximize the expected cumulative reward in T successive selections
- Which arm will you select?
- Optimal choice with complete information: select the arm with the highest mean reward
- How to select arms if the mean values are unknown a priori?

μ_1

μ_2

MAB Applications

- Online advertisement

MAB Applications

- Online advertisement
  - Display advertisements to maximize the clicks/revenue

MAB Applications

- Online advertisement
  - Display advertisements to maximize the clicks/revenue
  - Arms are advertisers

MAB Applications

- Online advertisement
  - Display advertisements to maximize the clicks/revenue
  - Arms are advertisers
  - Each arm has click-through-rate (probability of getting clicked)

MAB Applications

- Online advertisement
  - Display advertisements to maximize the clicks/revenue
  - Arms are advertisers
  - Each arm has click-through-rate (probability of getting clicked)
  - Platforms adaptively selects ads

MAB Applications

- Online advertisement
  - Display advertisements to maximize the clicks/revenue
  - Arms are advertisers
  - Each arm has click-through-rate (probability of getting clicked)
  - Platforms adaptively selects ads
  - Challenge: CTRs are not known

MAB Applications

- Online advertisement
  - Display advertisements to maximize the clicks/revenue
  - Arms are advertisers
  - Each arm has click-through-rate (probability of getting clicked)
  - Platforms adaptively selects ads
  - Challenge: CTRs are not known
- Recommendation systems

MAB Applications

- Online advertisement
  - Display advertisements to maximize the clicks/revenue
  - Arms are advertisers
  - Each arm has click-through-rate (probability of getting clicked)
  - Platforms adaptively selects ads
  - Challenge: CTRs are not known
- Recommendation systems
  - Netflix movie suggestions, Amazon product suggestions

MAB Applications

- Online advertisement
  - Display advertisements to maximize the clicks/revenue
  - Arms are advertisers
  - Each arm has click-through-rate (probability of getting clicked)
  - Platforms adaptively selects ads
  - Challenge: CTRs are not known
- Recommendation systems
  - Netflix movie suggestions, Amazon product suggestions
- Dynamic pricing

MAB Applications

- Online advertisement
  - Display advertisements to maximize the clicks/revenue
  - Arms are advertisers
  - Each arm has click-through-rate (probability of getting clicked)
  - Platforms adaptively selects ads
  - Challenge: CTRs are not known
- Recommendation systems
  - Netflix movie suggestions, Amazon product suggestions
- Dynamic pricing
- Rate adaptation in communication networks

MAB Applications

- Online advertisement
  - Display advertisements to maximize the clicks/revenue
  - Arms are advertisers
  - Each arm has click-through-rate (probability of getting clicked)
  - Platforms adaptively selects ads
  - Challenge: CTRs are not known
- Recommendation systems
  - Netflix movie suggestions, Amazon product suggestions
- Dynamic pricing
- Rate adaptation in communication networks
- Shortest path routing

MAB Applications

- Online advertisement
  - Display advertisements to maximize the clicks/revenue
  - Arms are advertisers
  - Each arm has click-through-rate (probability of getting clicked)
  - Platforms adaptively selects ads
  - Challenge: CTRs are not known
- Recommendation systems
  - Netflix movie suggestions, Amazon product suggestions
- Dynamic pricing
- Rate adaptation in communication networks
- Shortest path routing
- Clinical trials

MAB: Formulation

- Action at time t: $a_t \in \{1, \ldots, K\}$

MAB: Formulation

- Action at time t: $a_t \in \{1, \ldots, K\}$
- Reward at time t: $r(a_t) = \mathbb{1}\{k = a(t)\}X_i(t)$, where $X_k(t)$ is i.i.d. across arms and time

MAB: Formulation

- Action at time t: $a_t \in \{1, \ldots, K\}$
- Reward at time t: $r(a_t) = \mathbb{1}\{k = a(t)\} X_i(t)$, where $X_k(t)$ is i.i.d. across arms and time
- Assume that $\mu_k = \mathbb{E}[X_k(t)]$. Also, let $\mu_1 > \mu_2 > \ldots > \mu_K$

MAB: Formulation

- Action at time t: $a_t \in \{1, \ldots, K\}$
- Reward at time t: $r(a_t) = \mathbb{1}\{k = a(t)\}X_i(t)$, where $X_k(t)$ is i.i.d. across arms and time
- Assume that $\mu_k = \mathbb{E}[X_k(t)]$. Also, let $\mu_1 > \mu_2 > \ldots > \mu_K$
- Mean values are unknown

MAB: Formulation

- Action at time t: $a_t \in \{1, \ldots, K\}$
- Reward at time t: $r(a_t) = \mathbb{1}\{k = a(t)\}X_i(t)$, where $X_k(t)$ is i.i.d. across arms and time
- Assume that $\mu_k = \mathbb{E}[X_k(t)]$. Also, let $\mu_1 > \mu_2 > \ldots > \mu_K$
- Mean values are unknown
- Objective: select actions to maximize the expected cumulative reward

$$\max_{(a_t)_{t=1}^T} \quad \mathbb{E}[\sum_{t=1}^{T} r(a_t)]$$

Regret

- Optimal action with complete information $a^* = \arg\max_i \; \mu_i$

Regret

- Optimal action with complete information $a^* = \arg\max_i \ \mu_i$
  - Since $\mu_1 > \mu_2 > \ldots > \mu_K$, $a^* = 1$

Regret

- Optimal action with complete information $a^* = \arg\max_i \; \mu_i$
  - Since $\mu_1 > \mu_2 > \ldots > \mu_K$, $a^* = 1$
  - Maximum value of the expected cumulative reward is then $\mu_1 T$

Regret

- Optimal action with complete information $a^* = \arg\max_i \; \mu_i$
  - Since $\mu_1 > \mu_2 > \ldots > \mu_K$, $a^* = 1$
  - Maximum value of the expected cumulative reward is then $\mu_1 T$
- Regret of a learning algorithm

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)]$$

Regret

- Optimal action with complete information $a^* = \arg\max_i \ \mu_i$
  - Since $\mu_1 > \mu_2 > \ldots > \mu_K$, $a^* = 1$
  - Maximum value of the expected cumulative reward is then $\mu_1 T$
- Regret of a learning algorithm

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)]$$

  - Regret is the loss in cumulative reward due to lack of information

Regret

- Optimal action with complete information $a^* = \arg\max_i \; \mu_i$
  - Since $\mu_1 > \mu_2 > \ldots > \mu_K$, $a^* = 1$
  - Maximum value of the expected cumulative reward is then $\mu_1 T$
- Regret of a learning algorithm

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)]$$

  - Regret is the loss in cumulative reward due to lack of information
  - The first term indicates the performance of the optimal algorithm

Regret

- Optimal action with complete information $a^* = \arg\max_i \ \mu_i$
  - Since $\mu_1 > \mu_2 > \ldots > \mu_K$, $a^* = 1$
  - Maximum value of the expected cumulative reward is then $\mu_1 T$
- Regret of a learning algorithm

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)]$$

  - Regret is the loss in cumulative reward due to lack of information
  - The first term indicates the performance of the optimal algorithm
  - The second term indicates the performance of the learning algorithm

Regret

- Optimal action with complete information $a^* = \arg\max_i \ \mu_i$
  - Since $\mu_1 > \mu_2 > \ldots > \mu_K$, $a^* = 1$
  - Maximum value of the expected cumulative reward is then $\mu_1 T$
- Regret of a learning algorithm

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)]$$

  - Regret is the loss in cumulative reward due to lack of information
  - The first term indicates the performance of the optimal algorithm
  - The second term indicates the performance of the learning algorithm
- MAB objective: Find the learning algorithm that minimizes the regret

Regret

- Optimal action with complete information $a^* = \arg\max_i \ \mu_i$
  - Since $\mu_1 > \mu_2 > \ldots > \mu_K$, $a^* = 1$
  - Maximum value of the expected cumulative reward is then $\mu_1 T$
- Regret of a learning algorithm

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)]$$

  - Regret is the loss in cumulative reward due to lack of information
  - The first term indicates the performance of the optimal algorithm
  - The second term indicates the performance of the learning algorithm
- MAB objective: Find the learning algorithm that minimizes the regret
- Can we get a regret: $O(T^2)$? $O(T)$? $O(\sqrt{T})$? $O(\log T)$?

Regret

- Optimal action with complete information $a^* = \arg\max_i \ \mu_i$
  - Since $\mu_1 > \mu_2 > \ldots > \mu_K$, $a^* = 1$
  - Maximum value of the expected cumulative reward is then $\mu_1 T$
- Regret of a learning algorithm

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)]$$

  - Regret is the loss in cumulative reward due to lack of information
  - The first term indicates the performance of the optimal algorithm
  - The second term indicates the performance of the learning algorithm
- MAB objective: Find the learning algorithm that minimizes the regret
- Can we get a regret: $O(T^2)$? $O(T)$? $O(\sqrt{T})$? $O(\log T)$?
- What is the fundamental lower bound? Is there any algorithm that achieves the lower bound?

Regret Characterization

- Let $n_k(t)$ be th number of times arm k has been selected until t.

Regret Characterization

- Let $n_k(t)$ be th number of times arm k has been selected until t.

Regret Characterization

- Let $n_k(t)$ be th number of times arm k has been selected until t.

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)] = \mu_1 T - \mathbb{E}[\sum_{k=1}^{K} \mu_k n_k(T)] = \mathbb{E}[\sum_{k=1}^{K} (\mu_1 - \mu_k) n_k(T)]$$

Regret Characterization

- Let $n_k(t)$ be th number of times arm k has been selected until t.

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)] = \mu_1 T - \mathbb{E}[\sum_{k=1}^{K} \mu_k n_k(T)] = \mathbb{E}[\sum_{k=1}^{K} (\mu_1 - \mu_k) n_k(T)]$$

- We want $\mathbb{E}[n_k(T)]$ to be as small as possible for $k \neq 1$

Regret Characterization

- Let $n_k(t)$ be th number of times arm k has been selected until t.

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)] = \mu_1 T - \mathbb{E}[\sum_{k=1}^{K} \mu_k n_k(T)] = \mathbb{E}[\sum_{k=1}^{K} (\mu_1 - \mu_k) n_k(T)]$$

- We want $\mathbb{E}[n_k(T)]$ to be as small as possible for $k \neq 1$
- We want to get the best sublinear regret, i.e., $\lim_{T\to\infty} \frac{\text{Reg}(T)}{T} = 0$

Regret Characterization

- Let $n_k(t)$ be th number of times arm k has been selected until t.

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)] = \mu_1 T - \mathbb{E}[\sum_{k=1}^{K} \mu_k n_k(T)] = \mathbb{E}[\sum_{k=1}^{K} (\mu_1 - \mu_k) n_k(T)]$$

- We want $\mathbb{E}[n_k(T)]$ to be as small as possible for $k \neq 1$
- We want to get the best sublinear regret, i.e., $\lim_{T\to\infty} \frac{\text{Reg}(T)}{T} = 0$
  - Per step regret is converging to zero. Optimal in the asymptotic sense.

Regret Characterization

- Let $n_k(t)$ be th number of times arm k has been selected until t.

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)] = \mu_1 T - \mathbb{E}[\sum_{k=1}^{K} \mu_k n_k(T)] = \mathbb{E}[\sum_{k=1}^{K} (\mu_1 - \mu_k) n_k(T)]$$

- We want $\mathbb{E}[n_k(T)]$ to be as small as possible for $k \neq 1$
- We want to get the best sublinear regret, i.e., $\lim_{T\to\infty} \frac{\text{Reg}(T)}{T} = 0$
  - Per step regret is converging to zero. Optimal in the asymptotic sense.
  - Also, minimizes the cumulative cost of learning

Regret Characterization

- Let $n_k(t)$ be th number of times arm k has been selected until t.

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)] = \mu_1 T - \mathbb{E}[\sum_{k=1}^{K} \mu_k n_k(T)] = \mathbb{E}[\sum_{k=1}^{K} (\mu_1 - \mu_k) n_k(T)]$$

- We want $\mathbb{E}[n_k(T)]$ to be as small as possible for $k \neq 1$
- We want to get the best sublinear regret, i.e., $\lim_{T\to\infty} \frac{\text{Reg}(T)}{T} = 0$
  - Per step regret is converging to zero. Optimal in the asymptotic sense.
  - Also, minimizes the cumulative cost of learning

Regret Characterization

- Let $n_k(t)$ be th number of times arm k has been selected until t.

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)] = \mu_1 T - \mathbb{E}[\sum_{k=1}^{K} \mu_k n_k(T)] = \mathbb{E}[\sum_{k=1}^{K} (\mu_1 - \mu_k) n_k(T)]$$

- We want $\mathbb{E}[n_k(T)]$ to be as small as possible for $k \neq 1$
- We want to get the best sublinear regret, i.e., $\lim_{T\to\infty} \frac{\text{Reg}(T)}{T} = 0$
  - Per step regret is converging to zero. Optimal in the asymptotic sense.
  - Also, minimizes the cumulative cost of learning
- Regret of a greedy policy?

Regret Characterization

- Let $n_k(t)$ be th number of times arm k has been selected until t.

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)] = \mu_1 T - \mathbb{E}[\sum_{k=1}^{K} \mu_k n_k(T)] = \mathbb{E}[\sum_{k=1}^{K} (\mu_1 - \mu_k) n_k(T)]$$

- We want $\mathbb{E}[n_k(T)]$ to be as small as possible for $k \neq 1$
- We want to get the best sublinear regret, i.e., $\lim_{T\to\infty} \frac{\text{Reg}(T)}{T} = 0$
  - Per step regret is converging to zero. Optimal in the asymptotic sense.
  - Also, minimizes the cumulative cost of learning
- Regret of a greedy policy?
  - Greedy can lock onto a suboptimal action forever (linear regret)

Regret Characterization

- Let $n_k(t)$ be th number of times arm k has been selected until t.

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)] = \mu_1 T - \mathbb{E}[\sum_{k=1}^{K} \mu_k n_k(T)] = \mathbb{E}[\sum_{k=1}^{K} (\mu_1 - \mu_k) n_k(T)]$$

- We want $\mathbb{E}[n_k(T)]$ to be as small as possible for $k \neq 1$
- We want to get the best sublinear regret, i.e., $\lim_{T \to \infty} \frac{\text{Reg}(T)}{T} = 0$
  - Per step regret is converging to zero. Optimal in the asymptotic sense.
  - Also, minimizes the cumulative cost of learning
- Regret of a greedy policy?
  - Greedy can lock onto a suboptimal action forever (linear regret)
- Regret of an ϵ-greedy policy?

Regret Characterization

- Let $n_k(t)$ be th number of times arm k has been selected until t.

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)] = \mu_1 T - \mathbb{E}[\sum_{k=1}^{K} \mu_k n_k(T)] = \mathbb{E}[\sum_{k=1}^{K} (\mu_1 - \mu_k) n_k(T)]$$

- We want $\mathbb{E}[n_k(T)]$ to be as small as possible for $k \neq 1$
- We want to get the best sublinear regret, i.e., $\lim_{T\to\infty} \frac{\text{Reg}(T)}{T} = 0$
  - Per step regret is converging to zero. Optimal in the asymptotic sense.
  - Also, minimizes the cumulative cost of learning
- Regret of a greedy policy?
  - Greedy can lock onto a suboptimal action forever (linear regret)
- Regret of an ϵ-greedy policy?
  - Explore always, with the same epsilon probability (linear regret)

Regret Characterization

- Let $n_k(t)$ be th number of times arm k has been selected until t.

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)] = \mu_1 T - \mathbb{E}[\sum_{k=1}^{K} \mu_k n_k(T)] = \mathbb{E}[\sum_{k=1}^{K} (\mu_1 - \mu_k) n_k(T)]$$

- We want $\mathbb{E}[n_k(T)]$ to be as small as possible for $k \neq 1$
- We want to get the best sublinear regret, i.e., $\lim_{T\to\infty} \frac{\text{Reg}(T)}{T} = 0$
  - Per step regret is converging to zero. Optimal in the asymptotic sense.
  - Also, minimizes the cumulative cost of learning
- Regret of a greedy policy?
  - Greedy can lock onto a suboptimal action forever (linear regret)
- Regret of an ϵ-greedy policy?
  - Explore always, with the same epsilon probability (linear regret)
- Regret of an ϵ_t-greedy policy?

Regret Characterization

- Let $n_k(t)$ be th number of times arm k has been selected until t.

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)] = \mu_1 T - \mathbb{E}[\sum_{k=1}^{K} \mu_k n_k(T)] = \mathbb{E}[\sum_{k=1}^{K} (\mu_1 - \mu_k) n_k(T)]$$

- We want $\mathbb{E}[n_k(T)]$ to be as small as possible for $k \neq 1$
- We want to get the best sublinear regret, i.e., $\lim_{T\to\infty} \frac{\text{Reg}(T)}{T} = 0$
  - Per step regret is converging to zero. Optimal in the asymptotic sense.
  - Also, minimizes the cumulative cost of learning
- Regret of a greedy policy?
  - Greedy can lock onto a suboptimal action forever (linear regret)
- Regret of an ϵ-greedy policy?
  - Explore always, with the same epsilon probability (linear regret)
- Regret of an ϵ_t-greedy policy?
  - Can get sublinear regret. But need additional information

Regret: Fundamental Lower Bound

- Fundamental lower bound [Lai and Robbins, 1985]: Assume that $X_k(t) \sim P_k$ (i.e., P_k is the distribution of the rewards from arm k). Then,

$$\liminf_{T\to\infty} \frac{\mathsf{Reg}(T)}{\log T} \geq \sum_{k=2}^{K} \frac{(\mu_1 - \mu_k)}{D_{KL}(P_k, P_1)}$$

Regret: Fundamental Lower Bound

- Fundamental lower bound [Lai and Robbins, 1985]: Assume that $X_k(t) \sim P_k$ (i.e., P_k is the distribution of the rewards from arm k). Then,

$$\liminf_{T \to \infty} \frac{\text{Reg}(T)}{\log T} \geq \sum_{k=2}^{K} \frac{(\mu_1 - \mu_k)}{D_{KL}(P_k, P_1)}$$

- Fundamental regret lower bound is $\Omega(K \log T)$

Regret: Fundamental Lower Bound

- Fundamental lower bound [Lai and Robbins, 1985]: Assume that $X_k(t) \sim P_k$ (i.e., P_k is the distribution of the rewards from arm k). Then,

$$\liminf_{T\to\infty} \frac{\text{Reg}(T)}{\log T} \geq \sum_{k=2}^{K} \frac{(\mu_1 - \mu_k)}{D_{KL}(P_k, P_1)}$$

- Fundamental regret lower bound is $\Omega(K \log T)$
- Asymptotic regret of any learning algorithm cannot be smaller than $O(K \log T)$

Regret: Fundamental Lower Bound

- Fundamental lower bound [Lai and Robbins, 1985]: Assume that $X_k(t) \sim P_k$ (i.e., P_k is the distribution of the rewards from arm k). Then,

$$\liminf_{T\to\infty} \frac{\text{Reg}(T)}{\log T} \geq \sum_{k=2}^{K} \frac{(\mu_1 - \mu_k)}{D_{KL}(P_k, P_1)}$$

- Fundamental regret lower bound is $\Omega(K \log T)$
- Asymptotic regret of any learning algorithm cannot be smaller than $O(K \log T)$
- Can we develop an algorithm that achieves this regret performance?

Upper Confidence Bound (UCB) Algorithm

UCB algorithm

- Let $\hat{\mu}_k(t) = \frac{1}{n_k(t)} \sum_{\tau=1}^{t} r(a_\tau)\, \mathbb{I}\{a_\tau = k\}$ be the empirical mean reward obtained from arm k until t

UCB algorithm

- Let $\hat{\mu}_k(t) = \frac{1}{n_k(t)} \sum_{\tau=1}^{t} r(a_\tau)\, \mathbb{I}\{a_\tau = k\}$ be the empirical mean reward obtained from arm k until t
- UCB Index: $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{2 \log t}{n_k(t)}}$

UCB algorithm

- Let $\hat{\mu}_k(t) = \frac{1}{n_k(t)} \sum_{\tau=1}^{t} r(a_\tau) \, \mathbb{I}\{a_\tau = k\}$ be the empirical mean reward obtained from arm k until t
- UCB Index: $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{2 \log t}{n_k(t)}}$
- UCB Algorithm: Select the arm with the highest index, i.e., $a(t) = \arg\max_k \; g_k(t)$

UCB algorithm

- Let $\hat{\mu}_k(t) = \frac{1}{n_k(t)} \sum_{\tau=1}^{t} r(a_\tau)\, \mathbb{I}\{a_\tau = k\}$ be the empirical mean reward obtained from arm k until t
- UCB Index: $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{2 \log t}{n_k(t)}}$
- UCB Algorithm: Select the arm with the highest index, i.e., $a(t) = \arg\max_k \; g_k(t)$
- UCB index:

$$g_{k,t} = \underbrace{\hat{\mu}_k(t)}_{\text{exploitation term}} + \underbrace{\sqrt{\frac{2 \log t}{n_k(t)}}}_{\text{exploration term (ucb term)}}$$

UCB algorithm

- Let $\hat{\mu}_k(t) = \frac{1}{n_k(t)} \sum_{\tau=1}^{t} r(a_\tau)\, \mathbb{I}\{a_\tau = k\}$ be the empirical mean reward obtained from arm k until t
- UCB Index: $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{2\log t}{n_k(t)}}$
- UCB Algorithm: Select the arm with the highest index, i.e., $a(t) = \arg\max_k \; g_k(t)$
- UCB index:

$$g_{k,t} = \underbrace{\hat{\mu}_k(t)}_{\text{exploitation term}} + \underbrace{\sqrt{\frac{2\log t}{n_k(t)}}}_{\text{exploration term (ucb term)}}$$

- Exploitation term forces to select arm with the best empirical performance so far (greedy)

UCB algorithm

- Let $\hat{\mu}_k(t) = \frac{1}{n_k(t)} \sum_{\tau=1}^{t} r(a_\tau)\ \mathbb{I}\{a_\tau = k\}$ be the empirical mean reward obtained from arm k until t
- UCB Index: $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{2\log t}{n_k(t)}}$
- UCB Algorithm: Select the arm with the highest index, i.e., $a(t) = \arg\max_k \ g_k(t)$
- UCB index:

$$g_{k,t} = \underbrace{\hat{\mu}_k(t)}_{\text{exploitation term}} + \underbrace{\sqrt{\frac{2\log t}{n_k(t)}}}_{\text{exploration term (ucb term)}}$$

- Exploitation term forces to select arm with the best empirical performance so far (greedy)
- Exploration term forces to select arms that has not been select enough

Performance of UCB Algorithm

Theorem (Auer et al, 2002)

Under UCB algorithm,

$$\mathrm{Reg}(T) \leq 8 \sum_{k=2}^{K} \frac{\log T}{(\mu_1 - \mu_k)} + 4K$$

Performance of UCB Algorithm

Theorem (Auer et al, 2002)

Under UCB algorithm,

$$\text{Reg}(T) \leq 8 \sum_{k=2}^{K} \frac{\log T}{(\mu_1 - \mu_k)} + 4K$$

- Regret of UCB algorithm is $O(K \log T)$

Performance of UCB Algorithm

Theorem (Auer et al, 2002)

Under UCB algorithm,

$$\text{Reg}(T) \leq 8 \sum_{k=2}^{K} \frac{\log T}{(\mu_1 - \mu_k)} + 4K$$

- Regret of UCB algorithm is $O(K \log T)$
- Recall that lower bound on the regret for any algorithm is $\Omega(K \log T)$

Performance of UCB Algorithm

Theorem (Auer et al, 2002)

Under UCB algorithm,

$$\text{Reg}(T) \leq 8 \sum_{k=2}^{K} \frac{\log T}{(\mu_1 - \mu_k)} + 4K$$

- Regret of UCB algorithm is $O(K \log T)$
- Recall that lower bound on the regret for any algorithm is $\Omega(K \log T)$
- So, UCB algorithm is order optimal

Concentration Inequality

- Concentration inequality: A concentration inequality provides bounds on how a random variable deviates from some value (typically, its expected value)

Concentration Inequality

- Concentration inequality: A concentration inequality provides bounds on how a random variable deviates from some value (typically, its expected value)
- Usually, concentration inequalities are used to characterize the convergence rate of empirical mean

Concentration Inequality

- Concentration inequality: A concentration inequality provides bounds on how a random variable deviates from some value (typically, its expected value)
- Usually, concentration inequalities are used to characterize the convergence rate of empirical mean
- More precisely, let $\hat{\mu}(t) = \frac{1}{t}\sum_{\tau=1}^{t} X_\tau$, where X_τs are i.i.d. random variables with $\mathbb{E}[X_\tau] = \mu$

Concentration Inequality

- Concentration inequality: A concentration inequality provides bounds on how a random variable deviates from some value (typically, its expected value)
- Usually, concentration inequalities are used to characterize the convergence rate of empirical mean
- More precisely, let $\hat{\mu}(t) = \frac{1}{t}\sum_{\tau=1}^{t} X_\tau$, where X_τs are i.i.d. random variables with $\mathbb{E}[X_\tau] = \mu$
- We want to characterize $\mathbb{P}(|\hat{\mu}(t) - \mu| \geq \epsilon)$ as a function of t

Concentration Inequality

- Concentration inequality: A concentration inequality provides bounds on how a random variable deviates from some value (typically, its expected value)
- Usually, concentration inequalities are used to characterize the convergence rate of empirical mean
- More precisely, let $\hat{\mu}(t) = \frac{1}{t}\sum_{\tau=1}^{t} X_\tau$, where X_τs are i.i.d. random variables with $\mathbb{E}[X_\tau] = \mu$
- We want to characterize $\mathbb{P}(|\hat{\mu}(t) - \mu| \geq \epsilon)$ as a function of t

Lemma (Hoeffding's inequality)

Let $\hat{\mu}(t) = \frac{1}{t}\sum_{\tau=1}^{t} X_\tau$, where X_τs are i.i.d. random variables with $\mathbb{E}[X_\tau] = \mu$. Also, let $b_l \leq X_\tau \leq b_h$. Then,

$$\mathbb{P}(|\hat{\mu}(t) - \mu| \geq \epsilon) \leq 2 \exp\left(-\frac{2t\epsilon^2}{(b_h - b_l)^2}\right)$$

Proof of the UCB Algorithm - 1

- Instead of using $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{2\log t}{n_k(t)}}$, we will use a simplified index for proof, $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{\log(1/\delta)}{2n_k(t)}}$

Proof of the UCB Algorithm - 1

- Instead of using $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{2\log t}{n_k(t)}}$, we will use a simplified index for proof, $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{\log(1/\delta)}{2n_k(t)}}$
- Also, denote $g_{k,m_k} = \hat{\mu}_{k,m_k} + \sqrt{\frac{\log(1/\delta)}{2m_k}}$

Proof of the UCB Algorithm - 1

- Instead of using $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{2\log t}{n_k(t)}}$, we will use a simplified index for proof, $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{\log(1/\delta)}{2n_k(t)}}$
- Also, denote $g_{k,m_k} = \hat{\mu}_{k,m_k} + \sqrt{\frac{\log(1/\delta)}{2m_k}}$
- Recall that $\text{Reg}(T) = \mathbb{E}[\sum_{k=1}^{K} \Delta_k n_k(T)]$, where $\Delta_k = (\mu_1 - \mu_k)$

Proof of the UCB Algorithm - 1

- Instead of using $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{2\log t}{n_k(t)}}$, we will use a simplified index for proof, $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{\log(1/\delta)}{2n_k(t)}}$
- Also, denote $g_{k,m_k} = \hat{\mu}_{k,m_k} + \sqrt{\frac{\log(1/\delta)}{2m_k}}$
- Recall that $\text{Reg}(T) = \mathbb{E}[\sum_{k=1}^{K} \Delta_k n_k(T)]$, where $\Delta_k = (\mu_1 - \mu_k)$
- We will bound $\mathbb{E}[n_k(T)]$ and show that it is $O(\log T)$

Proof of the UCB Algorithm - 1

- Instead of using $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{2\log t}{n_k(t)}}$, we will use a simplified index for proof, $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{\log(1/\delta)}{2n_k(t)}}$
- Also, denote $g_{k,m_k} = \hat{\mu}_{k,m_k} + \sqrt{\frac{\log(1/\delta)}{2m_k}}$
- Recall that $\text{Reg}(T) = \mathbb{E}[\sum_{k=1}^{K} \Delta_k n_k(T)]$, where $\Delta_k = (\mu_1 - \mu_k)$
- We will bound $\mathbb{E}[n_k(T)]$ and show that it is $O(\log T)$
- Define the "good" event

$$G_k = \{\mu_1 \leq \min_t g_1(t)\} \cap \{g_{k,m_k} \leq \mu_1\}$$

Proof of the UCB Algorithm - 1

- Instead of using $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{2\log t}{n_k(t)}}$, we will use a simplified index for proof, $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{\log(1/\delta)}{2n_k(t)}}$
- Also, denote $g_{k,m_k} = \hat{\mu}_{k,m_k} + \sqrt{\frac{\log(1/\delta)}{2m_k}}$
- Recall that $\text{Reg}(T) = \mathbb{E}[\sum_{k=1}^{K} \Delta_k n_k(T)]$, where $\Delta_k = (\mu_1 - \mu_k)$
- We will bound $\mathbb{E}[n_k(T)]$ and show that it is $O(\log T)$
- Define the "good" event

$$G_k = \{\mu_1 \leq \min_t g_1(t)\} \cap \{g_{k,m_k} \leq \mu_1\}$$

- We can show that

Proof of the UCB Algorithm - 1

- Instead of using $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{2 \log t}{n_k(t)}}$, we will use a simplified index for proof, $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{\log(1/\delta)}{2n_k(t)}}$
- Also, denote $g_{k,m_k} = \hat{\mu}_{k,m_k} + \sqrt{\frac{\log(1/\delta)}{2m_k}}$
- Recall that $\text{Reg}(T) = \mathbb{E}[\sum_{k=1}^{K} \Delta_k n_k(T)]$, where $\Delta_k = (\mu_1 - \mu_k)$
- We will bound $\mathbb{E}[n_k(T)]$ and show that it is $O(\log T)$
- Define the "good" event

$$G_k = \{\mu_1 \leq \min_t g_1(t)\} \cap \{g_{k,m_k} \leq \mu_1\}$$

- We can show that
  1. If the good event occurs, then arm k will be played at most m_k times: $n_k(T) \leq m_k$

Proof of the UCB Algorithm - 1

- Instead of using $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{2\log t}{n_k(t)}}$, we will use a simplified index for proof, $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{\log(1/\delta)}{2n_k(t)}}$
- Also, denote $g_{k,m_k} = \hat{\mu}_{k,m_k} + \sqrt{\frac{\log(1/\delta)}{2m_k}}$
- Recall that $\text{Reg}(T) = \mathbb{E}[\sum_{k=1}^{K} \Delta_k n_k(T)]$, where $\Delta_k = (\mu_1 - \mu_k)$
- We will bound $\mathbb{E}[n_k(T)]$ and show that it is $O(\log T)$
- Define the "good" event

$$G_k = \{\mu_1 \leq \min_t g_1(t)\} \cap \{g_{k,m_k} \leq \mu_1\}$$

- We can show that
  1. If the good event occurs, then arm k will be played at most m_k times: $n_k(T) \leq m_k$
  2. The complement event G_k^c occurs with low probability (depending on m_k)

Proof of the UCB Algorithm - 1

- Instead of using $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{2\log t}{n_k(t)}}$, we will use a simplified index for proof, $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{\log(1/\delta)}{2n_k(t)}}$
- Also, denote $g_{k,m_k} = \hat{\mu}_{k,m_k} + \sqrt{\frac{\log(1/\delta)}{2m_k}}$
- Recall that $\text{Reg}(T) = \mathbb{E}[\sum_{k=1}^{K} \Delta_k n_k(T)]$, where $\Delta_k = (\mu_1 - \mu_k)$
- We will bound $\mathbb{E}[n_k(T)]$ and show that it is $O(\log T)$
- Define the "good" event

$$G_k = \{\mu_1 \leq \min_t g_1(t)\} \cap \{g_{k,m_k} \leq \mu_1\}$$

- We can show that
  1. If the good event occurs, then arm k will be played at most m_k times: $n_k(T) \leq m_k$
  2. The complement event G_k^c occurs with low probability (depending on m_k)
- So, we can bound

$$\mathbb{E}[n_k(T)] = \mathbb{E}[n_k(T)\mathbb{1}\{G\}] + \mathbb{E}[n_k(T)\mathbb{1}\{G^c\}] \leq m_k + T\mathbb{P}(G^c)$$

Proof of the UCB Algorithm - 1

- Instead of using $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{2 \log t}{n_k(t)}}$, we will use a simplified index for proof, $g_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{\log(1/\delta)}{2n_k(t)}}$
- Also, denote $g_{k,m_k} = \hat{\mu}_{k,m_k} + \sqrt{\frac{\log(1/\delta)}{2m_k}}$
- Recall that $\text{Reg}(T) = \mathbb{E}[\sum_{k=1}^{K} \Delta_k n_k(T)]$, where $\Delta_k = (\mu_1 - \mu_k)$
- We will bound $\mathbb{E}[n_k(T)]$ and show that it is $O(\log T)$
- Define the "good" event

$$G_k = \{\mu_1 \leq \min_t g_1(t)\} \cap \{g_{k,m_k} \leq \mu_1\}$$

- We can show that
  1. If the good event occurs, then arm k will be played at most m_k times: $n_k(T) \leq m_k$
  2. The complement event G_k^c occurs with low probability (depending on m_k)
- So, we can bound

$$\mathbb{E}[n_k(T)] = \mathbb{E}[n_k(T)\mathbb{1}\{G\}] + \mathbb{E}[n_k(T)\mathbb{1}\{G^c\}] \leq m_k + T\mathbb{P}(G^c)$$

- We will choose $m_k = C \log T$ and show that under this choice, $T\mathbb{P}(G^c)$ is a constant

Proof of the UCB Algorithm - 2

- We will bound the probability of $G_k^c = \{\mu_1 \leq \min_t g_1(t)\}^c \cup \left\{g_{k,m_k} \leq \mu_1\right\}^c$

Proof of the UCB Algorithm - 2

- We will bound the probability of $G_k^c = \{\mu_1 \leq \min_t g_1(t)\}^c \cup \left\{g_{k,m_k} \leq \mu_1\right\}^c$
- Consider the first set:

$$\{\mu_1 \geq \min_t g_1(t)\} \subset \{\mu_1 \geq \min_t g_{1,t}\} = \cup_t \{\mu_1 \geq g_{1,t}\}$$

Proof of the UCB Algorithm - 2

- We will bound the probability of $G_k^c = \{\mu_1 \leq \min_t g_1(t)\}^c \cup \left\{g_{k,m_k} \leq \mu_1\right\}^c$
- Consider the first set:

$$\{\mu_1 \geq \min_t g_1(t)\} \subset \{\mu_1 \geq \min_t g_{1,t}\} = \cup_t \{\mu_1 \geq g_{1,t}\}$$

- So, we can bound the probability as

$$\begin{aligned}\mathbb{P}(\{\mu_1 \geq \min_t g_1(t)\}) \leq \mathbb{P}(\{\mu_1 \geq \min_t g_{1,t}\}) = \mathbb{P}(\cup_t \{\mu_1 \geq g_{1,t}\}) \\ \leq \sum_{t=1}^{T} \mathbb{P}(\mu_1 \geq g_{1,t}) = \sum_{t=1}^{T} \mathbb{P}(\mu_1 \geq \hat{\mu}_{1,t} + \sqrt{\frac{\log(1/\delta)}{2t}})\end{aligned}$$

Proof of the UCB Algorithm - 2

- We will bound the probability of $G_k^c = \{\mu_1 \le \min_t g_1(t)\}^c \cup \{g_{k,m_k} \le \mu_1\}^c$
- Consider the first set:

$$\{\mu_1 \ge \min_t g_1(t)\} \subset \{\mu_1 \ge \min_t g_{1,t}\} = \cup_t\{\mu_1 \ge g_{1,t}\}$$

- So, we can bound the probability as

$$\begin{aligned}\mathbb{P}(\{\mu_1 \ge \min_t g_1(t)\}) &\le \mathbb{P}(\{\mu_1 \ge \min_t g_{1,t}\}) = \mathbb{P}(\cup_t\{\mu_1 \ge g_{1,t}\}) \\ &\le \sum_{t=1}^{T} \mathbb{P}(\mu_1 \ge g_{1,t}) = \sum_{t=1}^{T} \mathbb{P}(\mu_1 \ge \hat{\mu}_{1,t} + \sqrt{\frac{\log(1/\delta)}{2t}})\end{aligned}$$

- Using Hoeffding inequality

$$\mathbb{P}(\mu_1 \ge \hat{\mu}_{k,t} + \sqrt{\frac{\log(1/\delta)}{2t}}) \le 2\text{exp}\left(-2t\left(\sqrt{\frac{\log(1/\delta)}{2t}}\right)^2\right) = 2\text{exp}(-\log(1/\delta)) = 2\delta$$

Proof of the UCB Algorithm - 2

- We will bound the probability of $G_k^c = \{\mu_1 \le \min_t g_1(t)\}^c \cup \{g_{k,m_k} \le \mu_1\}^c$
- Consider the first set:

$$\{\mu_1 \ge \min_t g_1(t)\} \subset \{\mu_1 \ge \min_t g_{1,t}\} = \cup_t \{\mu_1 \ge g_{1,t}\}$$

- So, we can bound the probability as

$$\mathbb{P}(\{\mu_1 \ge \min_t g_1(t)\}) \le \mathbb{P}(\{\mu_1 \ge \min_t g_{1,t}\}) = \mathbb{P}(\cup_t \{\mu_1 \ge g_{1,t}\})$$
$$\le \sum_{t=1}^{T} \mathbb{P}(\mu_1 \ge g_{1,t}) = \sum_{t=1}^{T} \mathbb{P}(\mu_1 \ge \hat{\mu}_{1,t} + \sqrt{\frac{\log(1/\delta)}{2t}})$$

- Using Hoeffding inequality

$$\mathbb{P}(\mu_1 \ge \hat{\mu}_{k,t} + \sqrt{\frac{\log(1/\delta)}{2t}}) \le 2\text{exp}\left(-2t\left(\sqrt{\frac{\log(1/\delta)}{2t}}\right)^2\right) = 2\text{exp}(-\log(1/\delta)) = 2\delta$$

- So, we can bound the probability of the fist set as

$$\mathbb{P}(\{\mu_1 \ge \min_t g_1(t)\}) \le 2T\delta$$

Proof of the UCB Algorithm - 3

- We will bound the probability of $G_k^c = \{\mu_1 \leq \min_t g_1(t)\}^c \cup \left\{g_{k,m_k} \leq \mu_1\right\}^c$

Proof of the UCB Algorithm - 3

- We will bound the probability of $G_k^c = \{\mu_1 \leq \min_t g_1(t)\}^c \cup \left\{g_{k,m_k} \leq \mu_1\right\}^c$
- We got a bound for the first set. Now, we will consider the second set.

Proof of the UCB Algorithm - 3

- We will bound the probability of $G_k^c = \{\mu_1 \le \min_t g_1(t)\}^c \cup \left\{g_{k,m_k} \le \mu_1\right\}^c$
- We got a bound for the first set. Now, we will consider the second set.
- We first assume that m_k is large enough so that $\Delta_k - \sqrt{\frac{\log(1/\delta)}{2m_k}} \ge \frac{1}{2}\Delta_k$

Proof of the UCB Algorithm - 3

- We will bound the probability of $G_k^c = \{\mu_1 \le \min_t g_1(t)\}^c \cup \left\{g_{k,m_k} \le \mu_1\right\}^c$
- We got a bound for the first set. Now, we will consider the second set.
- We first assume that m_k is large enough so that $\Delta_k - \sqrt{\frac{\log(1/\delta)}{2m_k}} \ge \frac{1}{2}\Delta_k$
- Note that this can be achieved by selecting $m_k \ge \frac{2\log(1/\delta)}{\Delta_k^2}$. Choose this m_k.

Proof of the UCB Algorithm - 3

- We will bound the probability of $G_k^c = \{\mu_1 \leq \min_t g_1(t)\}^c \cup \{g_{k,m_k} \leq \mu_1\}^c$
- We got a bound for the first set. Now, we will consider the second set.
- We first assume that m_k is large enough so that $\Delta_k - \sqrt{\frac{\log(1/\delta)}{2m_k}} \geq \frac{1}{2}\Delta_k$
- Note that this can be achieved by selecting $m_k \geq \frac{2\log(1/\delta)}{\Delta_k^2}$. Choose this m_k.
- Now, we can bound the probability of the second set as

$$\mathbb{P}(\{g_{k,m_k} \geq \mu_1\}) = \mathbb{P}(\{\hat{\mu}_{k,m_k} + \sqrt{\frac{\log(1/\delta)}{2m_k}} \geq \mu_1\}) = \mathbb{P}(\{\hat{\mu}_{k,m_k} - \mu_k + \sqrt{\frac{\log(1/\delta)}{2m_k}} \geq \mu_1 - \mu_k\})$$

$$= \mathbb{P}(\{\hat{\mu}_{k,m_k} - \mu_k \geq \Delta_k - \sqrt{\frac{\log(1/\delta)}{2m_k}}\}) \leq \mathbb{P}(\{\hat{\mu}_{k,m_k} - \mu_k \geq \frac{\Delta_k}{2}\})$$

$$\leq 2\text{exp}(-2m_k\frac{\Delta_k^2}{4}) = 2\text{exp}(-\log(1/\delta)) = 2\delta$$

Proof of the UCB Algorithm - 4

- We will bound $G_k^c = \{\mu_1 \leq \min_t g_1(t)\}^c \cup \left\{g_{k,m_k} \leq \mu_1\right\}^c$

Proof of the UCB Algorithm - 4

- We will bound $G_k^c = \{\mu_1 \le \min_t g_1(t)\}^c \cup \left\{g_{k,m_k} \le \mu_1\right\}^c$
- Combining the probability bound for the first and second term, we get

$$\mathbb{P}(G_k^c) \le 2T\delta + 2\delta = 2(T+1)\delta$$

Proof of the UCB Algorithm - 4

- We will bound $G_k^c = \{\mu_1 \le \min_t g_1(t)\}^c \cup \left\{g_{k,m_k} \le \mu_1\right\}^c$
- Combining the probability bound for the first and second term, we get

$$\mathbb{P}(G_k^c) \le 2T\delta + 2\delta = 2(T+1)\delta$$

- Recall that we had: $\mathbb{E}[n_k(T)] \le m_k + T\mathbb{P}(G^c)$

Proof of the UCB Algorithm - 4

- We will bound $G_k^c = \{\mu_1 \leq \min_t g_1(t)\}^c \cup \left\{g_{k,m_k} \leq \mu_1\right\}^c$
- Combining the probability bound for the first and second term, we get

$$\mathbb{P}(G_k^c) \leq 2T\delta + 2\delta = 2(T+1)\delta$$

- Recall that we had: $\mathbb{E}[n_k(T)] \leq m_k + T\mathbb{P}(G^c)$
- Substituting, we get

$$\mathbb{E}[n_k(T)] \leq \frac{2\log(1/\delta)}{\Delta_k^2} + 2T(T+1)\delta$$

Proof of the UCB Algorithm - 4

- We will bound $G_k^c = \{\mu_1 \leq \min_t g_1(t)\}^c \cup \left\{g_{k,m_k} \leq \mu_1\right\}^c$
- Combining the probability bound for the first and second term, we get

$$\mathbb{P}(G_k^c) \leq 2T\delta + 2\delta = 2(T+1)\delta$$

- Recall that we had: $\mathbb{E}[n_k(T)] \leq m_k + T\mathbb{P}(G^c)$
- Substituting, we get

$$\mathbb{E}[n_k(T)] \leq \frac{2\log(1/\delta)}{\Delta_k^2} + 2T(T+1)\delta$$

- Choose $\delta = 1/T^2$. Then, we get

$$\mathbb{E}[n_k(T)] \leq \frac{4\log(T)}{\Delta_k^2} + 4$$

Proof of the UCB Algorithm - 4

- We will bound $G_k^c = \{\mu_1 \leq \min_t g_1(t)\}^c \cup \left\{g_{k,m_k} \leq \mu_1\right\}^c$
- Combining the probability bound for the first and second term, we get

$$\mathbb{P}(G_k^c) \leq 2T\delta + 2\delta = 2(T+1)\delta$$

- Recall that we had: $\mathbb{E}[n_k(T)] \leq m_k + T\mathbb{P}(G^c)$
- Substituting, we get

$$\mathbb{E}[n_k(T)] \leq \frac{2\log(1/\delta)}{\Delta_k^2} + 2T(T+1)\delta$$

- Choose $\delta = 1/T^2$. Then, we get

$$\mathbb{E}[n_k(T)] \leq \frac{4\log(T)}{\Delta_k^2} + 4$$

- Now, using the definition of regret

$$\text{Reg}(T) = \sum_{k=1}^{K} \Delta_k \mathbb{E}[n_k(T)] \leq \sum_{k=1}^{K} \frac{4}{\Delta_k} \log(T) + 4K$$

Problem Independent Upperbound

- For UCB Algorithm we first proved that, $\mathbb{E}[n_k(T)] \leq \frac{4\log(T)}{\Delta_k^2} + 4$. Using, this, we concluded that $\text{Reg}(T) = \sum_{k=1}^{K} \Delta_k \mathbb{E}[n_k(T)] \leq \sum_{k=1}^{K} \frac{4}{\Delta_k} \log(T) + 4K$.

Problem Independent Upperbound

- For UCB Algorithm we first proved that, $\mathbb{E}[n_k(T)] \leq \frac{4\log(T)}{\Delta_k^2} + 4$. Using, this, we concluded that $\text{Reg}(T) = \sum_{k=1}^{K} \Delta_k \mathbb{E}[n_k(T)] \leq \sum_{k=1}^{K} \frac{4}{\Delta_k} \log(T) + 4K$.
- What if Δ_k is very very small?

Problem Independent Upperbound

- For UCB Algorithm we first proved that, $\mathbb{E}[n_k(T)] \leq \frac{4\log(T)}{\Delta_k^2} + 4$. Using, this, we concluded that $\text{Reg}(T) = \sum_{k=1}^K \Delta_k \mathbb{E}[n_k(T)] \leq \sum_{k=1}^K \frac{4}{\Delta_k}\log(T) + 4K$.
- What if Δ_k is very very small?
- We can bound it differently as follows

$$\begin{aligned}\text{Reg}(T) &= \sum_{k=1}^{K} \Delta_k \mathbb{E}[n_k(T)] = \sum_{k:\Delta_k \leq \Delta} \Delta_k \mathbb{E}[n_k(T)] + \sum_{k:\Delta_k \geq \Delta} \Delta_k \mathbb{E}[n_k(T)] \\ &= T\Delta + \sum_{k:\Delta_k \geq \Delta} \Delta_k \mathbb{E}[n_k(T)] \leq T\Delta + \sum_{k:\Delta_k \geq \Delta} \left(\frac{4\log(T)}{\Delta_k} + 4\Delta_k\right) \leq T\Delta + \frac{4K\log(T)}{\Delta} + 4K\end{aligned}$$

Problem Independent Upperbound

- For UCB Algorithm we first proved that, $\mathbb{E}[n_k(T)] \leq \frac{4\log(T)}{\Delta_k^2} + 4$. Using, this, we concluded that $\text{Reg}(T) = \sum_{k=1}^{K} \Delta_k \mathbb{E}[n_k(T)] \leq \sum_{k=1}^{K} \frac{4}{\Delta_k} \log(T) + 4K$.
- What if Δ_k is very very small?
- We can bound it differently as follows

$$\text{Reg}(T) = \sum_{k=1}^{K} \Delta_k \mathbb{E}[n_k(T)] = \sum_{k:\Delta_k \leq \Delta} \Delta_k \mathbb{E}[n_k(T)] + \sum_{k:\Delta_k \geq \Delta} \Delta_k \mathbb{E}[n_k(T)]$$

$$= T\Delta + \sum_{k:\Delta_k \geq \Delta} \Delta_k \mathbb{E}[n_k(T)] \leq T\Delta + \sum_{k:\Delta_k \geq \Delta} \left(\frac{4\log(T)}{\Delta_k} + 4\Delta_k\right) \leq T\Delta + \frac{4K\log(T)}{\Delta} + 4K$$

Problem Independent Upperbound

- For UCB Algorithm we first proved that, $\mathbb{E}[n_k(T)] \leq \frac{4\log(T)}{\Delta_k^2} + 4$. Using, this, we concluded that $\text{Reg}(T) = \sum_{k=1}^{K} \Delta_k \mathbb{E}[n_k(T)] \leq \sum_{k=1}^{K} \frac{4}{\Delta_k} \log(T) + 4K$.
- What if Δ_k is very very small?
- We can bound it differently as follows

$$\begin{aligned}\text{Reg}(T) &= \sum_{k=1}^{K} \Delta_k \mathbb{E}[n_k(T)] = \sum_{k:\Delta_k \leq \Delta} \Delta_k \mathbb{E}[n_k(T)] + \sum_{k:\Delta_k \geq \Delta} \Delta_k \mathbb{E}[n_k(T)] \\ &= T\Delta + \sum_{k:\Delta_k \geq \Delta} \Delta_k \mathbb{E}[n_k(T)] \leq T\Delta + \sum_{k:\Delta_k \geq \Delta} (\frac{4\log(T)}{\Delta_k} + 4\Delta_k) \leq T\Delta + \frac{4K\log(T)}{\Delta} + 4K\end{aligned}$$

Choose $\Delta = \sqrt{\frac{16K\log T}{T}}$. Then, we get $\text{Reg}(T) \leq 5\sqrt{KT\log T} + 4K$

Problem Independent Upperbound

- For UCB Algorithm we first proved that, $\mathbb{E}[n_k(T)] \leq \frac{4\log(T)}{\Delta_k^2} + 4$. Using, this, we concluded that $\text{Reg}(T) = \sum_{k=1}^{K} \Delta_k \mathbb{E}[n_k(T)] \leq \sum_{k=1}^{K} \frac{4}{\Delta_k}\log(T) + 4K$.
- What if Δ_k is very very small?
- We can bound it differently as follows

$$\text{Reg}(T) = \sum_{k=1}^{K} \Delta_k \mathbb{E}[n_k(T)] = \sum_{k:\Delta_k \leq \Delta} \Delta_k \mathbb{E}[n_k(T)] + \sum_{k:\Delta_k \geq \Delta} \Delta_k \mathbb{E}[n_k(T)]$$

$$= T\Delta + \sum_{k:\Delta_k \geq \Delta} \Delta_k \mathbb{E}[n_k(T)] \leq T\Delta + \sum_{k:\Delta_k \geq \Delta} \left(\frac{4\log(T)}{\Delta_k} + 4\Delta_k\right) \leq T\Delta + \frac{4K\log(T)}{\Delta} + 4K$$

Choose $\Delta = \sqrt{\frac{16K\log T}{T}}$. Then, we get $\text{Reg}(T) \leq 5\sqrt{KT\log T} + 4K$

- Regret is $O(\sqrt{KT})$. Compare this with $O(\frac{K}{\Delta}\log T)$. What is the actual regret?

Problem Independent Upperbound

- For UCB Algorithm we first proved that, $\mathbb{E}[n_k(T)] \leq \frac{4\log(T)}{\Delta_k^2} + 4$. Using, this, we concluded that $\text{Reg}(T) = \sum_{k=1}^{K} \Delta_k \mathbb{E}[n_k(T)] \leq \sum_{k=1}^{K} \frac{4}{\Delta_k} \log(T) + 4K$.
- What if Δ_k is very very small?
- We can bound it differently as follows

$$\text{Reg}(T) = \sum_{k=1}^{K} \Delta_k \mathbb{E}[n_k(T)] = \sum_{k:\Delta_k \leq \Delta} \Delta_k \mathbb{E}[n_k(T)] + \sum_{k:\Delta_k \geq \Delta} \Delta_k \mathbb{E}[n_k(T)]$$

$$= T\Delta + \sum_{k:\Delta_k \geq \Delta} \Delta_k \mathbb{E}[n_k(T)] \leq T\Delta + \sum_{k:\Delta_k \geq \Delta} (\frac{4\log(T)}{\Delta_k} + 4\Delta_k) \leq T\Delta + \frac{4K\log(T)}{\Delta} + 4K$$

Choose $\Delta = \sqrt{\frac{16K\log T}{T}}$. Then, we get $\text{Reg}(T) \leq 5\sqrt{KT\log T} + 4K$

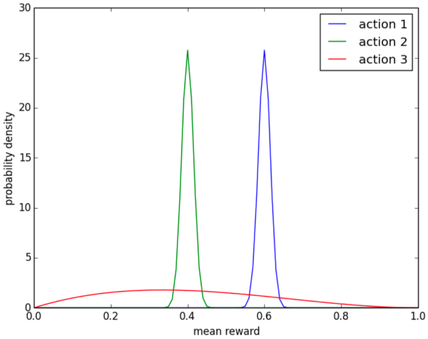
- Regret is $O(\sqrt{KT})$. Compare this with $O(\frac{K}{\Delta}\log T)$. What is the actual regret?
- We can actually write

$$\text{Reg}(T) = \min\{\frac{4K}{\Delta_{\min}} \log T,\ 5\sqrt{KT}\}$$

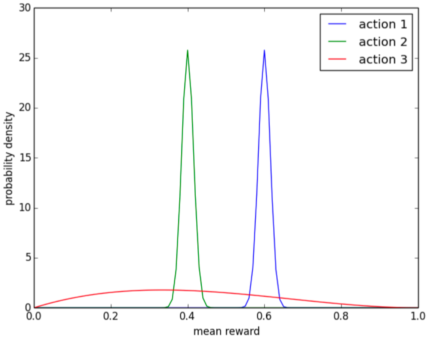
Posterior Sampling Algorithm

Bandit Algorithm using Posterior Sampling

- Which arm will you select?



Bandit Algorithm using Posterior Sampling

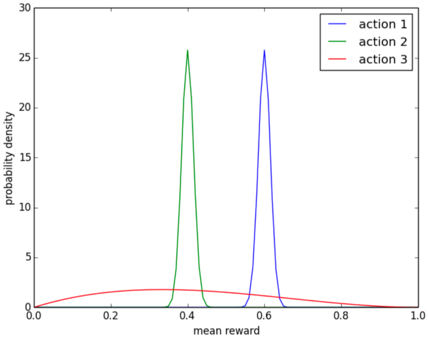


- Which arm will you select?

- Can we select an arm k proportional to the probability that μ_k is the maximum?

Bandit Algorithm using Posterior Sampling

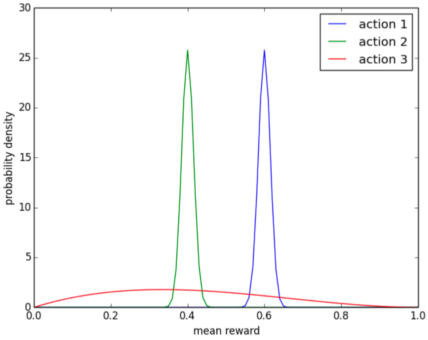
- Which arm will you select?



- Can we select an arm k proportional to the probability that μ_k is the maximum?
  - This randomization may induce enough exploration

Bandit Algorithm using Posterior Sampling

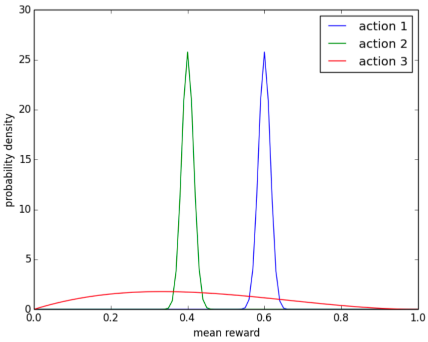
- Which arm will you select?



- Can we select an arm k proportional to the probability that μ_k is the maximum?
  - This randomization may induce enough exploration
- This probability is evaluated using the posterior distribution

Bandit Algorithm using Posterior Sampling

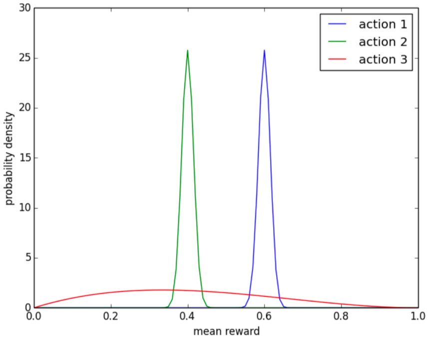
- Which arm will you select?



- Can we select an arm k proportional to the probability that μ_k is the maximum?
  - This randomization may induce enough exploration
- This probability is evaluated using the posterior distribution
- Exact probability matching is computationally challenging

Bandit Algorithm using Posterior Sampling

- Which arm will you select?



- Can we select an arm k proportional to the probability that μ_k is the maximum?
  - This randomization may induce enough exploration
- This probability is evaluated using the posterior distribution
- Exact probability matching is computationally challenging
- Posterior Sampling (also known as Thompson Sampling) overcomes this difficulty by sampling

Bayesian Inference: Prior, Likelihood, and Posterior

- Consider a random variable X with a distribution $p(\cdot|\theta)$, where θ is the parameter of the distribution

Bayesian Inference: Prior, Likelihood, and Posterior

- Consider a random variable X with a distribution $p(\cdot|\theta)$, where θ is the parameter of the distribution
- Assume a prior distribution p_{prior} on θ

Bayesian Inference: Prior, Likelihood, and Posterior

- Consider a random variable X with a distribution $p(\cdot|\theta)$, where θ is the parameter of the distribution
- Assume a prior distribution p_{prior} on θ
- Suppose we get one sample x_1 according to $p(\cdot|\theta)$

Bayesian Inference: Prior, Likelihood, and Posterior

- Consider a random variable X with a distribution $p(\cdot|\theta)$, where θ is the parameter of the distribution
- Assume a prior distribution p_{prior} on θ
- Suppose we get one sample x_1 according to $p(\cdot|\theta)$
- This sample provides some information about the unknown parameter θ

Bayesian Inference: Prior, Likelihood, and Posterior

- Consider a random variable X with a distribution $p(\cdot|\theta)$, where θ is the parameter of the distribution
- Assume a prior distribution p_{prior} on θ
- Suppose we get one sample x_1 according to $p(\cdot|\theta)$
- This sample provides some information about the unknown parameter θ
- We can use Baye's rule to update the distribution over θ

Bayesian Inference: Prior, Likelihood, and Posterior

- Consider a random variable X with a distribution $p(\cdot|\theta)$, where θ is the parameter of the distribution
- Assume a prior distribution p_{prior} on θ
- Suppose we get one sample x_1 according to $p(\cdot|\theta)$
- This sample provides some information about the unknown parameter θ
- We can use Baye's rule to update the distribution over θ

Bayesian Inference: Prior, Likelihood, and Posterior

- Consider a random variable X with a distribution $p(\cdot|\theta)$, where θ is the parameter of the distribution
- Assume a prior distribution p_{prior} on θ
- Suppose we get one sample x_1 according to $p(\cdot|\theta)$
- This sample provides some information about the unknown parameter θ
- We can use Baye's rule to update the distribution over θ

$$\mathbb{P}(\theta = y|x_1) = \frac{\mathbb{P}(x_1|\theta = y)\mathbb{P}(\theta = y)}{\mathbb{P}(x_1)} = \frac{\mathbb{P}(x_1|\theta = y)\mathbb{P}(\theta = y)}{\int_y \mathbb{P}(x_1|\theta = y)\mathbb{P}(\theta = y)dy}$$

Bayesian Inference: Prior, Likelihood, and Posterior

- Consider a random variable X with a distribution $p(\cdot|\theta)$, where θ is the parameter of the distribution
- Assume a prior distribution p_{prior} on θ
- Suppose we get one sample x_1 according to $p(\cdot|\theta)$
- This sample provides some information about the unknown parameter θ
- We can use Baye's rule to update the distribution over θ

$$\mathbb{P}(\theta = y|x_1) = \frac{\mathbb{P}(x_1|\theta = y)\mathbb{P}(\theta = y)}{\mathbb{P}(x_1)} = \frac{\mathbb{P}(x_1|\theta = y)\mathbb{P}(\theta = y)}{\int_y \mathbb{P}(x_1|\theta = y)\mathbb{P}(\theta = y)dy}$$

$$\mathbb{P}(\theta = y|x_1) \propto \mathbb{P}(x_1|\theta = y)\mathbb{P}(\theta = y)$$

Bayesian Inference: Prior, Likelihood, and Posterior

- Consider a random variable X with a distribution $p(\cdot|\theta)$, where θ is the parameter of the distribution
- Assume a prior distribution p_{prior} on θ
- Suppose we get one sample x_1 according to $p(\cdot|\theta)$
- This sample provides some information about the unknown parameter θ
- We can use Baye's rule to update the distribution over θ

$$\mathbb{P}(\theta = y|x_1) = \frac{\mathbb{P}(x_1|\theta = y)\mathbb{P}(\theta = y)}{\mathbb{P}(x_1)} = \frac{\mathbb{P}(x_1|\theta = y)\mathbb{P}(\theta = y)}{\int_y \mathbb{P}(x_1|\theta = y)\mathbb{P}(\theta = y)dy}$$

$$\mathbb{P}(\theta = y|x_1) \propto \mathbb{P}(x_1|\theta = y)\mathbb{P}(\theta = y)$$

- Prior, likelihood, and posterior

$$\underbrace{p(\theta = y|x_1)}_{\text{posterior}} \propto \underbrace{p(x_1|\theta = y)}_{\text{likelihood}} \underbrace{p_{\text{prior}}(\theta = y)}_{\text{prior}}$$

Bayesian Inference: Prior, Likelihood, and Posterior

- Prior, likelihood, and posterior

$$p(\theta = y|x_1) = \frac{p(x_1|\theta = y)\ p_{\text{prior}}(\theta = y)}{\int_y p(x_1|\theta = y)\ p_{\text{prior}}(\theta = y)dy}$$

Bayesian Inference: Prior, Likelihood, and Posterior

- Prior, likelihood, and posterior

$$p(\theta = y|x_1) = \frac{p(x_1|\theta = y)\; p_{\text{prior}}(\theta = y)}{\int_y p(x_1|\theta = y)\; p_{\text{prior}}(\theta = y) dy}$$

- In general, finding posterior distribution is computationally challenging

Bayesian Inference: Prior, Likelihood, and Posterior

- Prior, likelihood, and posterior

$$p(\theta = y|x_1) = \frac{p(x_1|\theta = y)\ p_{\text{prior}}(\theta = y)}{\int_y p(x_1|\theta = y)\ p_{\text{prior}}(\theta = y)dy}$$

- In general, finding posterior distribution is computationally challenging
- However, under some conditions, it is analytically tractable

Bayesian Inference: Prior, Likelihood, and Posterior

- Prior, likelihood, and posterior

$$p(\theta = y|x_1) = \frac{p(x_1|\theta = y)\; p_{\text{prior}}(\theta = y)}{\int_y p(x_1|\theta = y)\; p_{\text{prior}}(\theta = y)dy}$$

- In general, finding posterior distribution is computationally challenging
- However, under some conditions, it is analytically tractable
- If the posterior distribution is in the same probability distribution family as the prior distribution, the prior and posterior are called conjugate distributions, and the prior is called a conjugate prior for the likelihood function

Bayesian Inference: Prior, Likelihood, and Posterior

- Prior, likelihood, and posterior

$$p(\theta = y|x_1) = \frac{p(x_1|\theta = y)\ p_{\text{prior}}(\theta = y)}{\int_y p(x_1|\theta = y)\ p_{\text{prior}}(\theta = y)dy}$$

- In general, finding posterior distribution is computationally challenging
- However, under some conditions, it is analytically tractable
- If the posterior distribution is in the same probability distribution family as the prior distribution, the prior and posterior are called conjugate distributions, and the prior is called a conjugate prior for the likelihood function
- For example, exponential families have conjugate priors

Bayesian Inference: Conjugate Priors

- Consider a MAB problem where the reward from each arm is Bernoulli

Bayesian Inference: Conjugate Priors

- Consider a MAB problem where the reward from each arm is Bernoulli
- Let θ be the Bernoulli parameter of the first arm. So, $p(R = 1|\theta = y) = y$.

Bayesian Inference: Conjugate Priors

- Consider a MAB problem where the reward from each arm is Bernoulli
- Let θ be the Bernoulli parameter of the first arm. So, $p(R = 1|\theta = y) = y$.
  - This is a useful model, for example in modeling advertisement placement based on click through rate

Bayesian Inference: Conjugate Priors

- Consider a MAB problem where the reward from each arm is Bernoulli
- Let θ be the Bernoulli parameter of the first arm. So, $p(R = 1|\theta = y) = y$.
  - This is a useful model, for example in modeling advertisement placement based on click through rate
- Assume the prior distribution on θ as Beta(α, β)

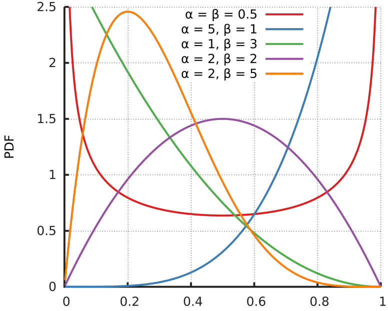
Bayesian Inference: Conjugate Priors

- Consider a MAB problem where the reward from each arm is Bernoulli
- Let θ be the Bernoulli parameter of the first arm. So, $p(R = 1|\theta = y) = y$.
  - This is a useful model, for example in modeling advertisement placement based on click through rate
- Assume the prior distribution on θ as Beta(α, β)

Bayesian Inference: Conjugate Priors

- Consider a MAB problem where the reward from each arm is Bernoulli
- Let θ be the Bernoulli parameter of the first arm. So, $p(R=1|\theta=y)=y$.
  - This is a useful model, for example in modeling advertisement placement based on click through rate
- Assume the prior distribution on θ as Beta(α,β)

$$p(\theta=y|(\alpha,\beta))=\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}y^{(\alpha-1)}(1-y)^{\beta-1}$$



Bayesian Inference: Conjugate Priors

- Consider a MAB problem where the reward from each arm is Bernoulli
- Let θ be the Bernoulli parameter of the first arm. So, $p(R = 1|\theta = y) = y$.
  - This is a useful model, for example in modeling advertisement placement based on click through rate
- Assume the prior distribution on θ as Beta(α, β)

Bayesian Inference: Conjugate Priors

- Consider a MAB problem where the reward from each arm is Bernoulli
- Let θ be the Bernoulli parameter of the first arm. So, $p(R = 1|\theta = y) = y$.
  - This is a useful model, for example in modeling advertisement placement based on click through rate
- Assume the prior distribution on θ as Beta(α, β)
- Suppose we get a reward $R = 1$. What is the posterior?

Bayesian Inference: Conjugate Priors

- Consider a MAB problem where the reward from each arm is Bernoulli
- Let θ be the Bernoulli parameter of the first arm. So, $p(R = 1|\theta = y) = y$.
  - This is a useful model, for example in modeling advertisement placement based on click through rate
- Assume the prior distribution on θ as Beta(α, β)
- Suppose we get a reward $R = 1$. What is the posterior?

$$\begin{aligned} p(\theta = y|R = 1) &\propto p(R = 1|\theta = y)\ p(\theta = y|(\alpha, \beta)) \\ &= y \ \ \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} y^{(\alpha-1)}(1-y)^{\beta-1} \\ &\propto y^{((\alpha+1)-1)}(1-y)^{\beta-1} \propto \text{Beta}(\alpha+1, \beta) \end{aligned}$$

Bayesian Inference: Conjugate Priors

- Consider a MAB problem where the reward from each arm is Bernoulli
- Let θ be the Bernoulli parameter of the first arm. So, $p(R = 1|\theta = y) = y$.
  - This is a useful model, for example in modeling advertisement placement based on click through rate
- Assume the prior distribution on θ as Beta(α, β)
- Suppose we get a reward $R = 1$. What is the posterior?

$$\begin{aligned} p(\theta = y|R = 1) &\propto p(R = 1|\theta = y)\ p(\theta = y|(\alpha, \beta)) \\ &= y \quad \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} y^{(\alpha-1)}(1-y)^{\beta-1} \\ &\propto y^{((\alpha+1)-1)}(1-y)^{\beta-1} \propto \text{Beta}(\alpha+1, \beta) \end{aligned}$$

- Similarly, if $R = 0$, posterior is Beta$(\alpha, \beta + 1)$

Bayesian Inference: Conjugate Priors

- Consider a MAB problem where the reward from each arm is Bernoulli
- Let θ be the Bernoulli parameter of the first arm. So, $p(R=1|\theta=y)=y$.
  - This is a useful model, for example in modeling advertisement placement based on click through rate
- Assume the prior distribution on θ as Beta(α,β)
- Suppose we get a reward $R=1$. What is the posterior?

$$\begin{aligned} p(\theta=y|R=1) &\propto p(R=1|\theta=y)\ p(\theta=y|(\alpha,\beta)) \\ &= y \quad \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} y^{(\alpha-1)}(1-y)^{\beta-1} \\ &\propto y^{((\alpha+1)-1)}(1-y)^{\beta-1} \propto \text{Beta}(\alpha+1,\beta) \end{aligned}$$

- Similarly, if $R=0$, posterior is Beta$(\alpha,\beta+1)$
- So, Bernoulli and Beta are conjugate priors

Posterior Sampling Algorithm for Bernoulli MAB

- Let $\text{Beta}(\alpha_k, \beta_k)$ be the posterior of arm k

Posterior Sampling Algorithm for Bernoulli MAB

- Let $\text{Beta}(\alpha_k, \beta_k)$ be the posterior of arm k
- We will initialize $\alpha_k = \beta_k = 1$ so that we start with uniform prior

Posterior Sampling Algorithm for Bernoulli MAB

- Let Beta(α_k, β_k) be the posterior of arm k
- We will initialize $\alpha_k = \beta_k = 1$ so that we start with uniform prior
- How do we select the next action (arm) a_t?

Posterior Sampling Algorithm for Bernoulli MAB

- Let Beta(α_k, β_k) be the posterior of arm k
- We will initialize $\alpha_k = \beta_k = 1$ so that we start with uniform prior
- How do we select the next action (arm) a_t?
- How do we update the posterior?

Posterior Sampling Algorithm for Bernoulli MAB

- Let Beta(α_k, β_k) be the posterior of arm k
- We will initialize $\alpha_k = \beta_k = 1$ so that we start with uniform prior
- How do we select the next action (arm) a_t?
- How do we update the posterior?

Posterior Sampling Algorithm for Bernoulli MAB

- Let $\text{Beta}(\alpha_k, \beta_k)$ be the posterior of arm k
- We will initialize $\alpha_k = \beta_k = 1$ so that we start with uniform prior
- How do we select the next action (arm) a_t?
- How do we update the posterior?

Algorithm 6 Posterior Sampling Algorithm

for $t = 1, 2, \ldots$ **do**
  # sample the parameter
  for $k = 1, \ldots, K$ **do**
    $\theta_k \sim \text{Beta}(\alpha_k, \beta_k)$
  end for
  # select action
  $a_t = \arg\max_k \ \theta_k$
  Observe the reward R_t
  # update the posterior
  $(\alpha_{a_t}, \beta_{a_t}) \leftarrow (\alpha_{a_t} + R_t, \beta_{a_t} + R_t)$
end for

Bayesian Regret

- Recall the regret definition

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)] = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} \sum_{k=1}^{K} \mathbb{1}\{a(t) = k\} X_k(t)]$$

Bayesian Regret

- Recall the regret definition

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)] = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T}\sum_{k=1}^{K} \mathbb{1}\{a(t) = k\} X_k(t)]$$

- We will make it more precise as follows:

Bayesian Regret

- Recall the regret definition

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)] = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T}\sum_{k=1}^{K} \mathbb{1}\{a(t) = k\} X_k(t)]$$

- We will make it more precise as follows:
- Let $\theta = (\theta_1, \theta_2, \ldots, \theta_K)$ be the parameter of the reward distribution, i.e., $X_k \sim p(\cdot|\theta_k)$. Denote $\mu^*(\theta) = \max_k\{\mu_k(\theta_k)\}$ where $\mu_k(\theta_k)$ is the mean reward from arm k.

Bayesian Regret

- Recall the regret definition

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)] = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T}\sum_{k=1}^{K} \mathbb{1}\{a(t) = k\} X_k(t)]$$

- We will make it more precise as follows:
- Let $\theta = (\theta_1, \theta_2, \ldots, \theta_K)$ be the parameter of the reward distribution, i.e., $X_k \sim p(\cdot|\theta_k)$. Denote $\mu^*(\theta) = \max_k\{\mu_k(\theta_k)\}$ where $\mu_k(\theta_k)$ is the mean reward from arm k.
- We can then define the regret when parameter is θ as

$$\text{Reg}(T;\theta) = \mu^* T - \mathbb{E}_{p(\cdot|\theta)}[\sum_{t=1}^{T} r(a_t)]$$

Bayesian Regret

- Recall the regret definition

$$\text{Reg}(T) = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T} r(a_t)] = \mu_1 T - \mathbb{E}[\sum_{t=1}^{T}\sum_{k=1}^{K} \mathbb{1}\{a(t) = k\} X_k(t)]$$

- We will make it more precise as follows:
- Let $\theta = (\theta_1, \theta_2, \ldots, \theta_K)$ be the parameter of the reward distribution, i.e., $X_k \sim p(\cdot|\theta_k)$. Denote $\mu^*(\theta) = \max_k\{\mu_k(\theta_k)\}$ where $\mu_k(\theta_k)$ is the mean reward from arm k.
- We can then define the regret when parameter is θ as

$$\text{Reg}(T;\theta) = \mu^* T - \mathbb{E}_{p(\cdot|\theta)}[\sum_{t=1}^{T} r(a_t)]$$

- Bayesian regret is defined as

$$\text{Bayes-Reg}(T) = \mathbb{E}_{\theta\sim p_{\text{prior}}}[\text{Reg}(T;\theta)]$$

Performance of Posterior Sampling Algorithm

Theorem

Under posterior sampling algorithm,

$$\textit{Reg}(T;\theta) = O\left(\frac{K}{\Delta_{\min}}\log T\right)$$

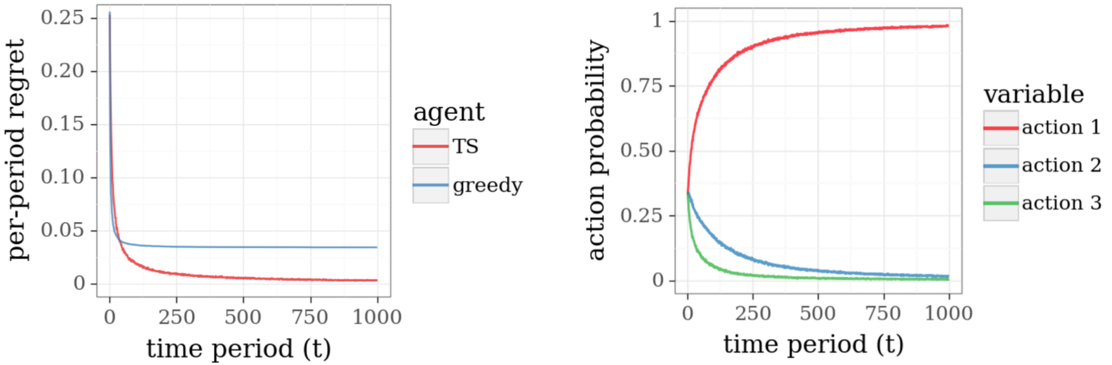
$$\textit{Bayes-Reg}(T) = \mathbb{E}_{\theta\sim p_{\text{prior}}}[\textit{Reg}(T;\theta)] = O(\sqrt{KT\log T})$$

$$\max_{\theta}\ \textit{Reg}(T;\theta) = O(\sqrt{KT\log T})$$

Performance of Posterior Sampling Algorithm

Theorem

Under posterior sampling algorithm, $\mathrm{Reg}(T) = O(K \log T)$



Exploration in Deep RL

Hard Exploration Problems

- An RL algorithm learns a policy using the reward feedback

Hard Exploration Problems

- An RL algorithm learns a policy using the reward feedback
- This works well if rewards are dense

Hard Exploration Problems

- An RL algorithm learns a policy using the reward feedback
- This works well if rewards are dense
- However, settings where rewards are sparse pose significant challenges for standard RL algorithms

Hard Exploration Problems

- An RL algorithm learns a policy using the reward feedback
- This works well if rewards are dense
- However, settings where rewards are sparse pose significant challenges for standard RL algorithms
- Many problems require long sequences of very specific actions to experience any reward. Such sequences are extremely unlikely to occur randomly

Hard Exploration Problems

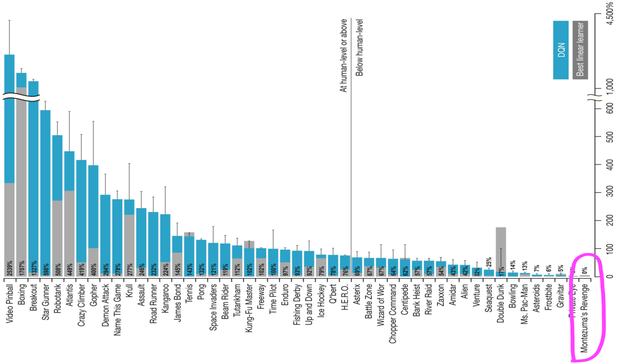
- An RL algorithm learns a policy using the reward feedback
- This works well if rewards are dense
- However, settings where rewards are sparse pose significant challenges for standard RL algorithms
- Many problems require long sequences of very specific actions to experience any reward. Such sequences are extremely unlikely to occur randomly
- Example: Montezuma's Revenge (an Atari game)

Hard Exploration Problems

- An RL algorithm learns a policy using the reward feedback
- This works well if rewards are dense
- However, settings where rewards are sparse pose significant challenges for standard RL algorithms
- Many problems require long sequences of very specific actions to experience any reward. Such sequences are extremely unlikely to occur randomly
- Example: Montezuma's Revenge (an Atari game)

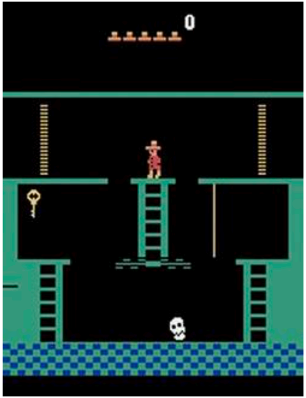
Hard Exploration Problems

- An RL algorithm learns a policy using the reward feedback
- This works well if rewards are dense
- However, settings where rewards are sparse pose significant challenges for standard RL algorithms
- Many problems require long sequences of very specific actions to experience any reward. Such sequences are extremely unlikely to occur randomly
- Example: Montezuma's Revenge (an Atari game)



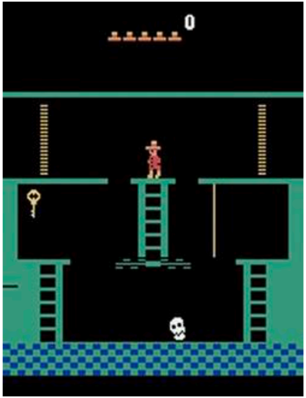
Exploration in Montezuma's Revenge

- There is reward for getting key and opening door
- No reward for getting killed by the skull (is it good? bad?)



Exploration in Montezuma's Revenge

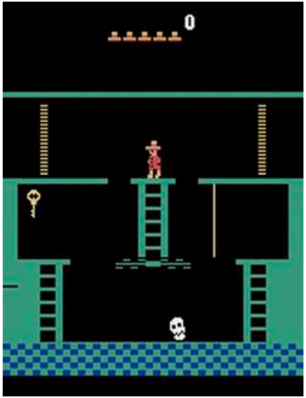
- There is reward for getting key and opening door
- No reward for getting killed by the skull (is it good? bad?)



- P(get key) = P(get down ladder 1) $\times$ P(get down rope) $\times$ P(get down ladder 2) $\times$ P(jump over skull) $\times$ P(get up ladder 3)

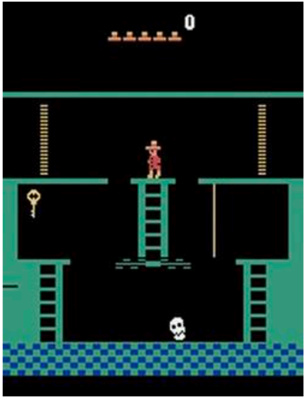
Exploration in Montezuma's Revenge

- There is reward for getting key and opening door
- No reward for getting killed by the skull (is it good? bad?)



- P(get key) = P(get down ladder 1) $\times$ P(get down rope) $\times$ P(get down ladder 2) $\times$ P(jump over skull) $\times$ P(get up ladder 3)
- Probability of getting a reward exponentially decreases with the length of the sequence

Exploration in Montezuma's Revenge



- There is reward for getting key and opening door
- No reward for getting killed by the skull (is it good? bad?)

- P(get key) = P(get down ladder 1) × P(get down rope) × P(get down ladder 2) × P(jump over skull) × P(get up ladder 3)
- Probability of getting a reward exponentially decreases with the length of the sequence
- Random exploration is unlikely to get a reward signal in such problems

Some References

- Count-based exploration
  - [Strehl and Littman, 2008, Bellemare et al., 2016, Tang et al., 2017]
- Prediction-based exploration
  - [Schmidhuber, 1991, Pathak et al., 2017, Burda et al., 2018a, Houthooft et al., 2016, Burda et al., 2018b]
- Memory-based exploration
  - [Badia et al., 2019, 2020]

- For a good survey of exploration in deep RL, see [Weng, 2020]

References I

Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

Marc G Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying count-based exploration and intrinsic motivation. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 1479–1487, 2016.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *31st Conference on Neural Information Processing Systems (NIPS)*, volume 30, pages 1–18, 2017.

Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pages 2778–2787. PMLR, 2017.

References II

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. In *International Conference on Learning Representations*, 2018a.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: variational information maximizing exploration. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 1117–1125, 2016.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018b.

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martin Arjovsky, Alexander Pritzel, Andrew Bolt, et al. Never give up: Learning directed exploration strategies. In *International Conference on Learning Representations*, 2019.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *International Conference on Machine Learning*, pages 507–517. PMLR, 2020.

References III

Lilian Weng. Exploration strategies in deep reinforcement learning. *lilianweng.github.io/lil-log*, 2020. URL https://lilianweng.github.io/lil-log/2020/06/07/exploration-strategies-in-deep-reinforcement-learning.html.