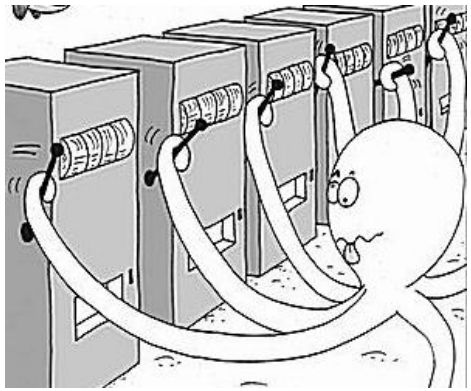
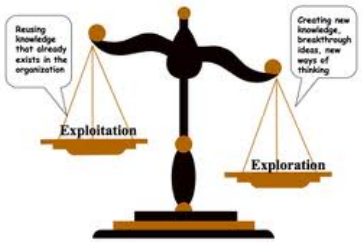
Web and Internet Economics

Multi–Armed Bandit



Matteo Papini



Motivating Example
Search Advertising

- Keyword allocations
- Assume 1 advertisement for each keyword search
- n possible ads: $a_1, a_2, \ldots, a_n$
- each ad a_i has a value v_i and a probability of being clicked p_i
- Search engine wants to maximize the expected revenue max $E\left[\sum_{t=1}^{T} R_t\right]$
- If p_i are known, choose a_{i^*}, where $i^* = arg\max_i p_i v_i$
- How to behave when p_i are unknown?

Motivating Example
Search Advertising

- Keyword allocations
- Assume 1 advertisement for each keyword search
- n possible ads: $a_1, a_2, \ldots, a_n$
- each ad a_i has a value v_i and a probability of being clicked p_i
- Search engine wants to maximize the expected revenue max $E\left[\sum_{t=1}^{T} R_t\right]$
- If p_i are known, choose a_{i^*}, where $i^* = arg\max_i p_i v_i$
- How to behave when p_i are unknown?

Motivating Example
Search Advertising

Marcello Restelli

- Keyword allocations
- Assume 1 advertisement for each keyword search
- n possible ads: $a_1, a_2, \ldots, a_n$
- each ad a_i has a value v_i and a probability of being clicked p_i
- Search engine wants to maximize the expected revenue max $E\left[\sum_{t=1}^{T} R_t\right]$
- If p_i are known, choose a_{i^*}, where $i^* = arg\max_i p_i v_i$
- How to behave when p_i are unknown?

Motivating Example
Search Advertising

- Keyword allocations
- Assume 1 advertisement for each keyword search
- n possible ads: $a_1, a_2, \ldots, a_n$
- each ad a_i has a value v_i and a probability of being clicked p_i
- Search engine wants to maximize the expected revenue max $E\left[\sum_{t=1}^{T} R_t\right]$
- If p_i are known, choose a_{i^*}, where $i^* = arg\max_i p_i v_i$
- How to behave when p_i are unknown?

Motivating Example
Search Advertising

- Keyword allocations
- Assume 1 advertisement for each keyword search
- n possible ads: $a_1, a_2, \ldots, a_n$
- each ad a_i has a value v_i and a probability of being clicked p_i
- Search engine wants to maximize the expected revenue max $E\left[\sum_{t=1}^{T} R_t\right]$
- If p_i are known, choose a_{i^*}, where $i^* = arg\max_i p_i v_i$
- How to behave when p_i are unknown?

Motivating Example
Search Advertising

- Keyword allocations
- Assume 1 advertisement for each keyword search
- n possible ads: $a_1, a_2, \ldots, a_n$
- each ad a_i has a value v_i and a probability of being clicked p_i
- Search engine wants to maximize the expected revenue max $E\left[\sum_{t=1}^{T} R_t\right]$
- If p_i are known, choose a_{i^*}, where $i^* = arg\max_i p_i v_i$
- How to behave when p_i are unknown?

Motivating Example
Search Advertising

Marcello Restelli

- Keyword allocations
- Assume 1 advertisement for each keyword search
- n possible ads: $a_1, a_2, \ldots, a_n$
- each ad a_i has a value v_i and a probability of being clicked p_i
- Search engine wants to maximize the expected revenue max $E\left[\sum_{t=1}^{T} R_t\right]$
- If p_i are known, choose a_{i^*}, where $i^* = arg\max_i p_i v_i$
- How to behave when p_i are unknown?

Exploration vs Exploitation Dilemma

- **Online** decision making involves a fundamental choice:
  - **Exploitation**: make the **best decision** given current information
  - **Exploration**: gather **more information**
- The best long–term strategy may involve **short–term sacrifices**
- Gather **enough information** to make the best overall decisions

Exploration vs Exploitation Dilemma

- **Online** decision making involves a fundamental choice:
  - **Exploitation**: make the **best decision** given current information
  - **Exploration**: gather **more information**
- The best long–term strategy may involve **short–term sacrifices**
- Gather **enough information** to make the best overall decisions

Exploration vs Exploitation Dilemma

- **Online** decision making involves a fundamental choice:
  - **Exploitation**: make the **best decision** given current information
  - **Exploration**: gather **more information**
- The best long–term strategy may involve **short–term sacrifices**
- Gather **enough information** to make the best overall decisions

Exploration vs Exploitation Dilemma

- **Online** decision making involves a fundamental choice:
  - **Exploitation**: make the **best decision** given current information
  - **Exploration**: gather **more information**
- The best long–term strategy may involve **short–term sacrifices**
- Gather **enough information** to make the best overall decisions

Exploration vs Exploitation Dilemma

- **Online** decision making involves a fundamental choice:
  - **Exploitation**: make the **best decision** given current information
  - **Exploration**: gather **more information**
- The best long–term strategy may involve **short–term sacrifices**
- Gather **enough information** to make the best overall decisions

Examples

- Online Banner Advertisements
  - Exploitation: Show the most successful advert
  - Exploration: Show a different advert
- Restaurant Selection
  - Exploitation: Go to favorite restaurant
  - Exploration: Try a new restaurant
- Oil Drilling
  - Exploitation: Drill at the best known location
  - Exploration: Drill at a new location
- Game Playing
  - Exploitation: Play the move you believe is best
  - Exploration: Play an experimental move
- Clinical Trial
  - Exploitation: Choose the best treatment so far
  - Exploration: Try a new treatment

Common Approaches in RL

- **ϵ–Greedy**

$$a_t = \begin{cases} a_t^* & \text{with probability } 1-\epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases}$$

- **Softmax**
  - **Bias** exploration towards promising actions
  - Softmax action selection methods **grade action probabilities** by estimated values
  - The most common softmax uses a **Gibbs (or Boltzmann) distribution**:

$$\pi(a|s) = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{a' \in \mathcal{A}} e^{\frac{Q(s,a')}{\tau}}}$$

  - τ is a **"computational" temperature**:
    - $\tau \to \infty$: uniform
    - $\tau \to 0$: greedy

Common Approaches in RL

- **ϵ–Greedy**

$$a_t = \begin{cases} a_t^* & \text{with probability } 1-\epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases}$$

- **Softmax**
  - **Bias** exploration towards promising actions
  - Softmax action selection methods **grade action probabilities** by estimated values
  - The most common softmax uses a **Gibbs (or Boltzmann) distribution**:

$$\pi(a|s) = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{a' \in \mathcal{A}} e^{\frac{Q(s,a')}{\tau}}}$$

  - τ is a **"computational" temperature**:
    - $\tau \to \infty$: $P = \frac{1}{|\mathcal{A}|}$
    - $\tau \to 0$: greedy

Common Approaches in RL

Marcello Restelli

- **ϵ–Greedy**

$$a_t = \begin{cases} a_t^* & \text{with probability } 1-\epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases}$$

- **Softmax**
  - **Bias** exploration towards promising actions
  - Softmax action selection methods **grade action probabilities** by estimated values
  - The most common softmax uses a **Gibbs (or Boltzmann) distribution**:

$$\pi(a|s) = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{a' \in \mathcal{A}} e^{\frac{Q(s,a')}{\tau}}}$$

  - τ is a **"computational" temperature**:
    - $\tau \to \infty$: $P = \frac{1}{|\mathcal{A}|}$
    - $\tau \to 0$: greedy

Common Approaches in RL

- **ϵ–Greedy**

$$a_t = \begin{cases} a_t^* & \text{with probability } 1-\epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases}$$

- **Softmax**
  - **Bias** exploration towards promising actions
  - Softmax action selection methods **grade action probabilities** by estimated values
  - The most common softmax uses a **Gibbs (or Boltzmann) distribution**:

$$\pi(a|s) = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{a' \in \mathcal{A}} e^{\frac{Q(s,a')}{\tau}}}$$

  - τ is a **"computational" temperature**:
    - $\tau \to \infty$: $P = \frac{1}{|\mathcal{A}|}$
    - $\tau \to 0$: greedy

Common Approaches in RL

- **ϵ–Greedy**

$$a_t = \begin{cases} a_t^* & \text{with probability } 1-\epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases}$$

- **Softmax**
  - **Bias** exploration towards promising actions
  - Softmax action selection methods **grade action probabilities** by estimated values
  - The most common softmax uses a **Gibbs (or Boltzmann) distribution**:

$$\pi(a|s) = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{a' \in \mathcal{A}} e^{\frac{Q(s,a')}{\tau}}}$$

  - τ is a **"computational" temperature**:
    - $\tau \to \infty$: $P = \frac{1}{|\mathcal{A}|}$
    - $\tau \to 0$: greedy

Common Approaches in RL

- **ϵ–Greedy**

$$a_t = \begin{cases} a_t^* & \text{with probability } 1-\epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases}$$

- **Softmax**
  - **Bias** exploration towards promising actions
  - Softmax action selection methods **grade action probabilities** by estimated values
  - The most common softmax uses a **Gibbs (or Boltzmann) distribution**:

$$\pi(a|s) = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{a' \in \mathcal{A}} e^{\frac{Q(s,a')}{\tau}}}$$

  - τ is a **"computational" temperature**:
    - $\tau \to \infty$: $P = \frac{1}{|\mathcal{A}|}$
    - $\tau \to 0$: greedy

Common Approaches in RL

- **ϵ–Greedy**

$$a_t = \begin{cases} a_t^* & \text{with probability } 1-\epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases}$$

- **Softmax**
  - **Bias** exploration towards promising actions
  - Softmax action selection methods **grade action probabilities** by estimated values
  - The most common softmax uses a **Gibbs (or Boltzmann) distribution**:

$$\pi(a|s) = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{a' \in \mathcal{A}} e^{\frac{Q(s,a')}{\tau}}}$$

  - τ is a **"computational" temperature**:
    - $\tau \to \infty$: $P = \frac{1}{|\mathcal{A}|}$
    - $\tau \to 0$: greedy

Common Approaches in RL

- **ϵ–Greedy**

$$a_t = \begin{cases} a_t^* & \text{with probability } 1-\epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases}$$

- **Softmax**
  - **Bias** exploration towards promising actions
  - Softmax action selection methods **grade action probabilities** by estimated values
  - The most common softmax uses a **Gibbs (or Boltzmann) distribution**:

$$\pi(a|s) = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{a' \in \mathcal{A}} e^{\frac{Q(s,a')}{\tau}}}$$

  - τ is a **"computational" temperature**:
    - $\tau \to \infty$: $P = \frac{1}{|\mathcal{A}|}$
    - $\tau \to 0$: greedy

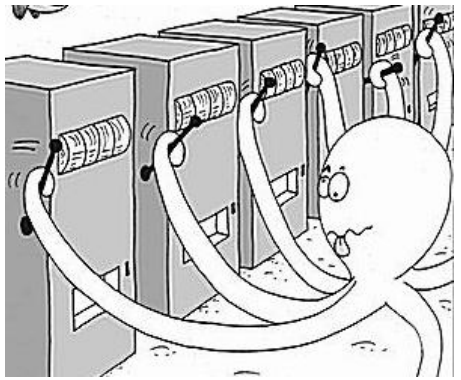
Outline

Marcello Restelli

Multi–Arm Bandits (MABs)

Marcello Restelli

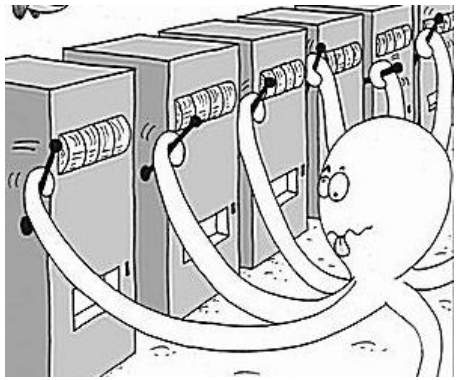
- A multi–armed bandit is a tuple $\langle \mathcal{A}, R \rangle$
- $\mathcal{A}$ is a set of N possible **actions** (one per machine = arm)
- $R(r|a)$ is an **unknown** probability distribution of rewards given the action chosen
- At each time step t the agent **selects** an action $a_t \in \mathcal{A}$
- The environment generates a **reward** $r_t \sim R(\cdot, a)$
- The **goal** is to maximize cumulative reward: $\sum_{t=1}^{T} r_t$



Multi–Arm Bandits (MABs)

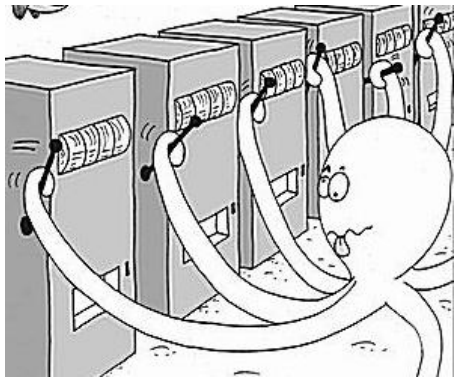
Marcello Restelli

- A multi–armed bandit is a tuple $\langle \mathcal{A}, R \rangle$
- $\mathcal{A}$ is a set of N possible **actions** (one per machine = arm)
- $R(r|a)$ is an **unknown** probability distribution of rewards given the action chosen
- At each time step t the agent **selects** an action $a_t \in \mathcal{A}$
- The environment generates a **reward** $r_t \sim R(\cdot, a)$
- The **goal** is to maximize cumulative reward: $\sum_{t=1}^{T} r_t$



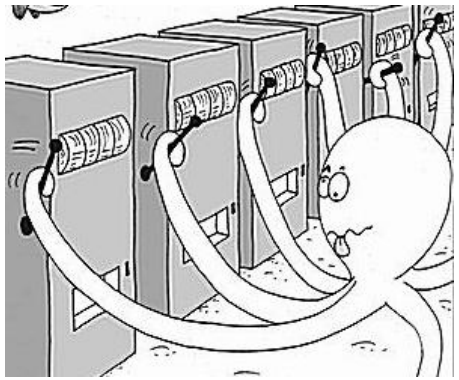
Multi–Arm Bandits (MABs)

- A multi–armed bandit is a tuple $\langle \mathcal{A}, R \rangle$
- $\mathcal{A}$ is a set of N possible **actions** (one per machine = arm)
- $R(r|a)$ is an **unknown** probability distribution of rewards given the action chosen
- At each time step t the agent **selects** an action $a_t \in \mathcal{A}$
- The environment generates a **reward** $r_t \sim R(\cdot, a)$
- The **goal** is to maximize cumulative reward: $\sum_{t=1}^{T} r_t$



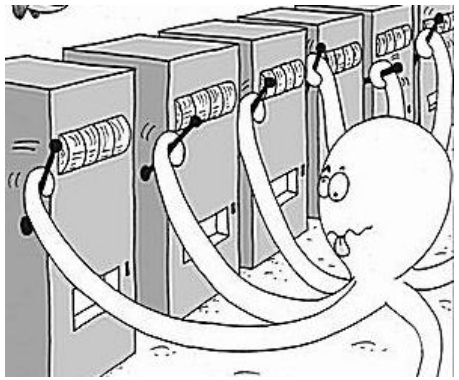
Multi–Arm Bandits (MABs)

- A multi–armed bandit is a tuple $\langle \mathcal{A}, R \rangle$
- $\mathcal{A}$ is a set of N possible **actions** (one per machine = arm)
- $R(r|a)$ is an **unknown** probability distribution of rewards given the action chosen
- At each time step t the agent **selects** an action $a_t \in \mathcal{A}$
- The environment generates a **reward** $r_t \sim R(\cdot, a)$
- The **goal** is to maximize cumulative reward: $\sum_{t=1}^{T} r_t$



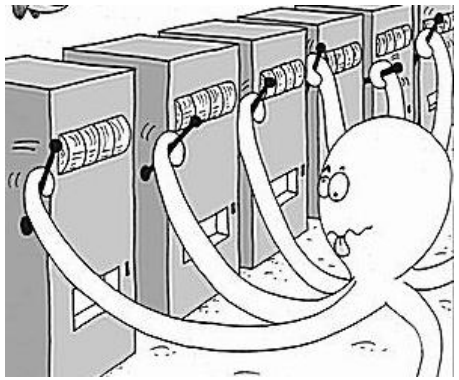
Multi–Arm Bandits (MABs)

- A multi–armed bandit is a tuple $\langle \mathcal{A}, R \rangle$
- $\mathcal{A}$ is a set of N possible **actions** (one per machine = arm)
- $R(r|a)$ is an **unknown** probability distribution of rewards given the action chosen
- At each time step t the agent **selects** an action $a_t \in \mathcal{A}$
- The environment generates a **reward** $r_t \sim R(\cdot, a)$
- The **goal** is to maximize cumulative reward: $\sum_{t=1}^{T} r_t$



Multi–Arm Bandits (MABs)

- A multi–armed bandit is a tuple $\langle \mathcal{A}, R \rangle$
- $\mathcal{A}$ is a set of N possible **actions** (one per machine = arm)
- $R(r|a)$ is an **unknown** probability distribution of rewards given the action chosen
- At each time step t the agent **selects** an action $a_t \in \mathcal{A}$
- The environment generates a **reward** $r_t \sim R(\cdot, a)$
- The **goal** is to maximize cumulative reward: $\sum_{t=1}^{T} r_t$



Regret

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

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

- The **optimal** value V^* is

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

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

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

- The **total regret** is the total opportunity loss

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

- Maximize cumulative reward $\equiv$ minimize total regret

Regret

Marcello Restelli

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

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

- The **optimal** value V^* is

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

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

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

- The **total regret** is the total opportunity loss

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

- Maximize cumulative reward $\equiv$ minimize total regret

Regret

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

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

- The **optimal** value V^* is

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

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

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

- The **total regret** is the total opportunity loss

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

- Maximize cumulative reward $\equiv$ minimize total regret

Regret

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

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

- The **optimal** value V^* is

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

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

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

- The **total regret** is the total opportunity loss

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

- Maximize cumulative reward $\equiv$ minimize total regret

Regret

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

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

- The **optimal** value V^* is

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

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

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

- The **total regret** is the total opportunity loss

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

- Maximize cumulative reward $\equiv$ minimize total regret

Counting Regret

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

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

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

Counting Regret

Marcello Restelli

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

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

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

Counting Regret

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

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

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

Counting Regret

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

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

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

Counting Regret

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

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

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

Greedy Algorithm

Marcello Restelli

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

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

- The **greedy** algorithm selects action with **highest value**

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

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

Greedy Algorithm

Marcello Restelli

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

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

- The **greedy** algorithm selects action with **highest value**

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

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

Greedy Algorithm

Marcello Restelli

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

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

- The **greedy** algorithm selects action with **highest value**

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

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

Greedy Algorithm

Marcello Restelli

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

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

- The **greedy** algorithm selects action with **highest value**

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

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

Greedy Algorithm

Marcello Restelli

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

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

- The **greedy** algorithm selects action with **highest value**

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

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

ϵ–Greedy Algorithm

Marcello Restelli

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

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

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

ϵ–Greedy Algorithm

Marcello Restelli

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

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

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

ϵ–Greedy Algorithm

Marcello Restelli

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

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

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

ϵ–Greedy Algorithm

Marcello Restelli

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

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

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

ϵ–Greedy Algorithm

Marcello Restelli

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

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

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

ϵ–Greedy on the 10–Armed Testbed

Marcello Restelli

- $N = 10$ possible actions
- $Q(a)$ are chosen randomly from a normal distribution $\mathcal{N}(0, 1)$
- Rewards r_t are also normal $\mathcal{N}(Q(a_t), 1)$
- 1000 plays
- Results averaged over 2000 trials

ϵ–Greedy on the 10–Armed Testbed

- $N = 10$ possible actions
- $Q(a)$ are chosen randomly from a normal distribution $\mathcal{N}(0, 1)$
- Rewards r_t are also normal $\mathcal{N}(Q(a_t), 1)$
- 1000 plays
- Results averaged over 2000 trials

ϵ–Greedy on the 10–Armed Testbed

Marcello Restelli

- $N = 10$ possible actions
- $Q(a)$ are chosen randomly from a normal distribution $\mathcal{N}(0, 1)$
- Rewards r_t are also normal $\mathcal{N}(Q(a_t), 1)$
- 1000 plays
- Results averaged over 2000 trials

ϵ–Greedy on the 10–Armed Testbed

Marcello Restelli

- $N = 10$ possible actions
- $Q(a)$ are chosen randomly from a normal distribution $\mathcal{N}(0, 1)$
- Rewards r_t are also normal $\mathcal{N}(Q(a_t), 1)$
- 1000 plays
- Results averaged over 2000 trials

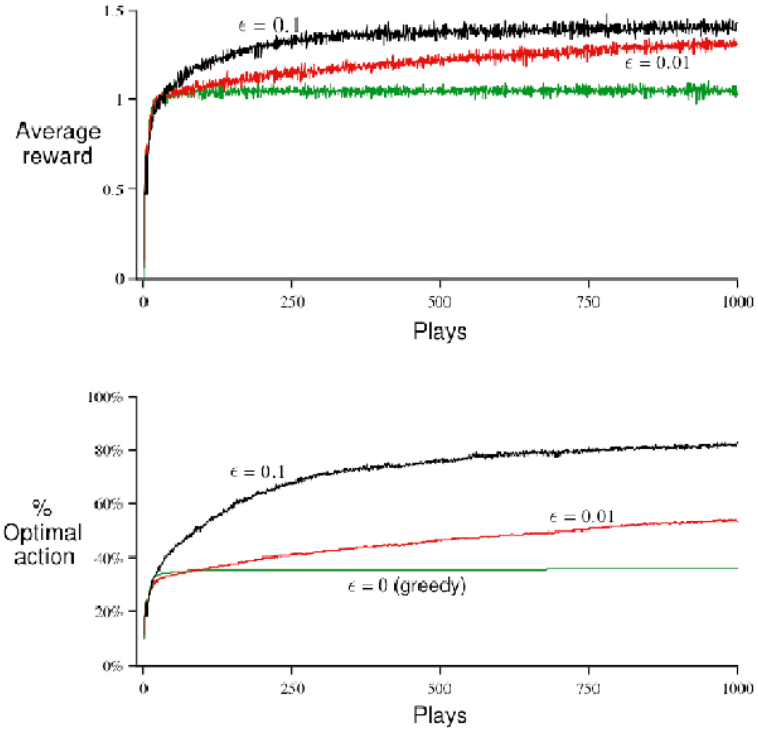
ϵ–Greedy on the 10–Armed Testbed

Marcello Restelli

- $N = 10$ possible actions
- $Q(a)$ are chosen randomly from a normal distribution $\mathcal{N}(0, 1)$
- Rewards r_t are also normal $\mathcal{N}(Q(a_t), 1)$
- 1000 plays
- Results averaged over 2000 trials

ϵ–Greedy on the 10–Armed Testbed

- $N = 10$ possible actions
- $Q(a)$ are chosen randomly from a normal distribution $\mathcal{N}(0, 1)$
- Rewards r_t are also normal $\mathcal{N}(Q(a_t), 1)$
- 1000 plays
- Results averaged over 2000 trials



Optimistic Initial Values

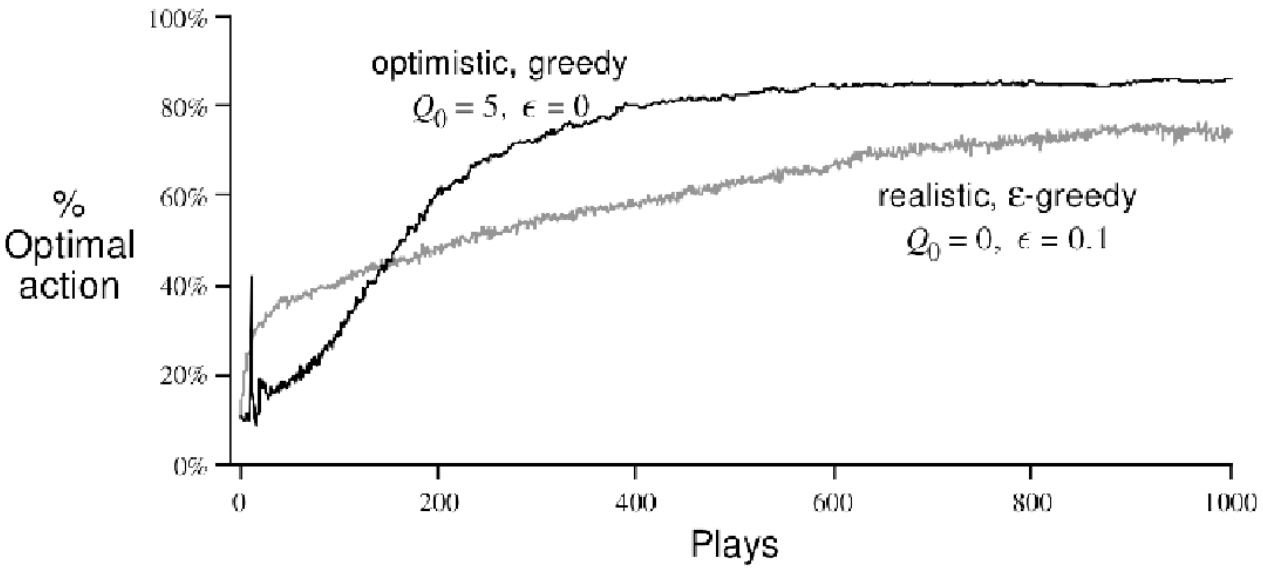
- **All** methods depend on $Q_0(a)$, i.e., they are **biased**
- Encourage exploration: **initialize** action values **optimistically**

Optimistic Initial Values

- **All** methods depend on $Q_0(a)$, i.e., they are **biased**
- Encourage exploration: **initialize** action values **optimistically**

Optimistic Initial Values

- **All** methods depend on $Q_0(a)$, i.e., they are **biased**
- Encourage exploration: **initialize** action values **optimistically**



Decaying ϵ_t–Greedy Algorithm

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

$$
\begin{aligned}
c &> 0 \\
d &= \min_{a|\Delta_a>0} \Delta_a \\
\epsilon_t &= \min\left\{1, \frac{c|\mathcal{A}|}{d^2 t}\right\}
\end{aligned}
$$

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

Decaying ϵ_t–Greedy Algorithm

Marcello Restelli

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

$$
\begin{aligned}
c &> 0 \\
d &= \min_{a|\Delta_a>0} \Delta_a \\
\epsilon_t &= \min\left\{1, \frac{c|\mathcal{A}|}{d^2 t}\right\}
\end{aligned}
$$

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

Decaying ϵ_t–Greedy Algorithm

Marcello Restelli

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

$$
\begin{aligned}
c &> 0 \\
d &= \min_{a|\Delta_a>0} \Delta_a \\
\epsilon_t &= \min\left\{1, \frac{c|\mathcal{A}|}{d^2 t}\right\}
\end{aligned}
$$

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

Decaying ϵ_t–Greedy Algorithm

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

$$
\begin{aligned}
c &> 0 \\
d &= \min_{a|\Delta_a>0} \Delta_a \\
\epsilon_t &= \min\left\{1, \frac{c|\mathcal{A}|}{d^2 t}\right\}
\end{aligned}
$$

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

Decaying ϵ_t–Greedy Algorithm

Marcello Restelli

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

$$
\begin{aligned}
c &> 0 \\
d &= \min_{a|\Delta_a>0} \Delta_a \\
\epsilon_t &= \min\left\{1, \frac{c|\mathcal{A}|}{d^2 t}\right\}
\end{aligned}
$$

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

Two Formulations

Marcello Restelli

- **Frequentist** formulation
  - $(Q(a_1), Q(a_2), \dots)$ are **unknown deterministic parameters**
  - **Policy**: choose an arm based on the observation history
  - **Total** reward over a **finite** horizon of length T
- **Bayesian** formulation
  - $(Q(a_1), Q(a_2), \dots)$ are random variables with **prior** distribution $(f_1, f_2, \dots)$
  - **Policy**: choose an arm based on the priors $(f_1, f_2, \dots)$ and the observation history

Two Formulations

- **Frequentist** formulation
  - $(Q(a_1), Q(a_2), \dots)$ are **unknown deterministic parameters**
  - **Policy**: choose an arm based on the observation history
  - **Total** reward over a **finite** horizon of length T
- **Bayesian** formulation
  - $(Q(a_1), Q(a_2), \dots)$ are random variables with **prior** distribution $(f_1, f_2, \dots)$
  - **Policy**: choose an arm based on the priors $(f_1, f_2, \dots)$ and the observation history

Two Formulations

Marcello Restelli

- **Frequentist** formulation
  - $(Q(a_1), Q(a_2), \dots)$ are **unknown deterministic parameters**
  - **Policy**: choose an arm based on the observation history
  - **Total** reward over a **finite** horizon of length T
- **Bayesian** formulation
  - $(Q(a_1), Q(a_2), \dots)$ are random variables with **prior** distribution $(f_1, f_2, \dots)$
  - **Policy**: choose an arm based on the priors $(f_1, f_2, \dots)$ and the observation history

Two Formulations

- **Frequentist** formulation
  - $(Q(a_1), Q(a_2), \dots)$ are **unknown deterministic parameters**
  - **Policy**: choose an arm based on the observation history
  - **Total** reward over a **finite** horizon of length T
- **Bayesian** formulation
  - $(Q(a_1), Q(a_2), \dots)$ are random variables with **prior** distribution $(f_1, f_2, \dots)$
  - **Policy**: choose an arm based on the priors $(f_1, f_2, \dots)$ and the observation history

Two Formulations

Marcello Restelli

- **Frequentist** formulation
  - $(Q(a_1), Q(a_2), \dots)$ are **unknown deterministic parameters**
  - **Policy**: choose an arm based on the observation history
  - **Total** reward over a **finite** horizon of length T
- **Bayesian** formulation
  - $(Q(a_1), Q(a_2), \dots)$ are random variables with **prior** distribution $(f_1, f_2, \dots)$
  - **Policy**: choose an arm based on the priors $(f_1, f_2, \dots)$ and the observation history

Two Formulations

- **Frequentist** formulation
  - $(Q(a_1), Q(a_2), \dots)$ are **unknown deterministic parameters**
  - **Policy**: choose an arm based on the observation history
  - **Total** reward over a **finite** horizon of length T
- **Bayesian** formulation
  - $(Q(a_1), Q(a_2), \dots)$ are random variables with **prior** distribution $(f_1, f_2, \dots)$
  - **Policy**: choose an arm based on the priors $(f_1, f_2, \dots)$ and the observation history

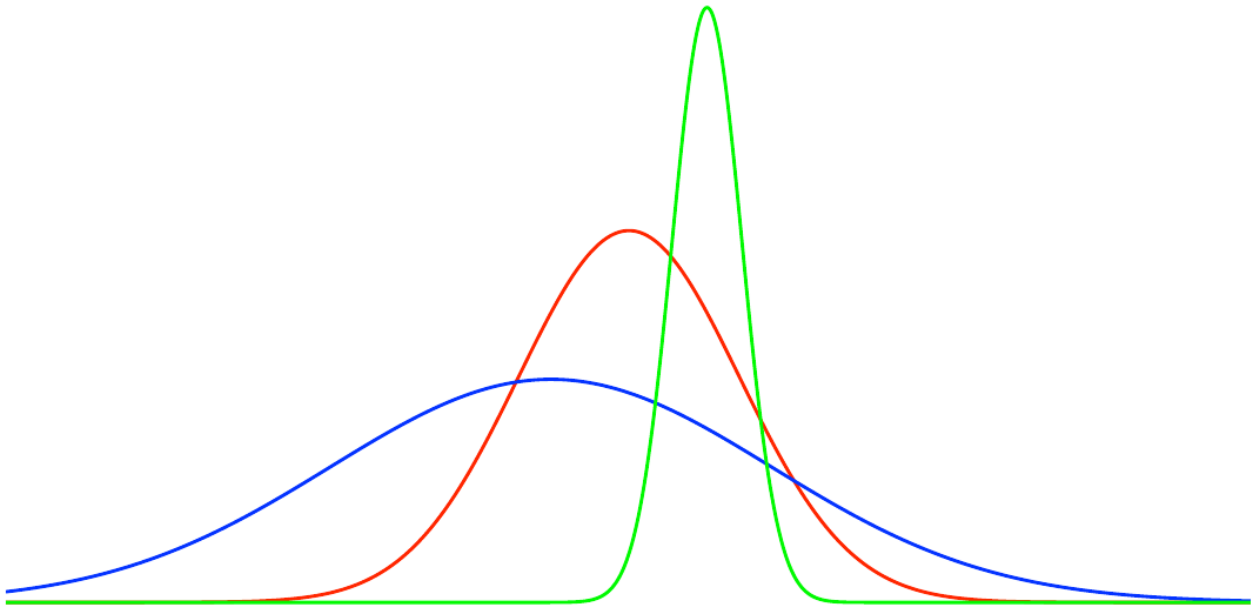
Two Formulations

- **Frequentist** formulation
  - $(Q(a_1), Q(a_2), \dots)$ are **unknown deterministic parameters**
  - **Policy**: choose an arm based on the observation history
  - **Total** reward over a **finite** horizon of length T
- **Bayesian** formulation
  - $(Q(a_1), Q(a_2), \dots)$ are random variables with **prior** distribution $(f_1, f_2, \dots)$
  - **Policy**: choose an arm based on the priors $(f_1, f_2, \dots)$ and the observation history

Outline

Optimism in Face of Uncertainty

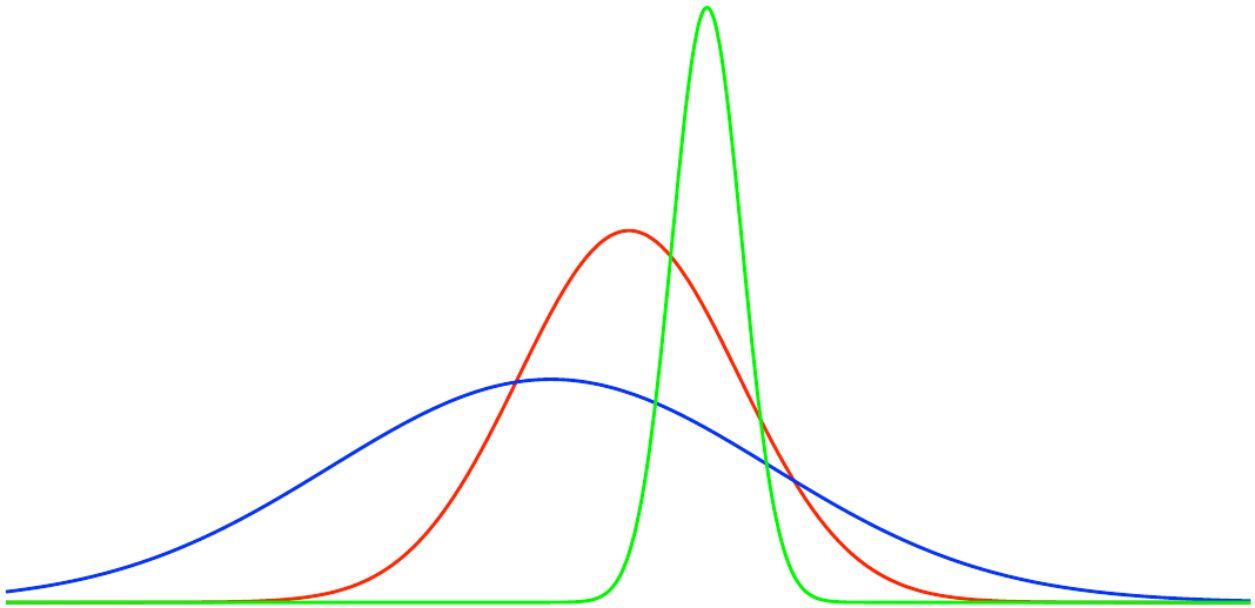
Marcello Restelli



- The more **uncertain** we are about an action–value
- The more important it is to **explore** that action
- It could turn out to be the **best action**

Optimism in Face of Uncertainty

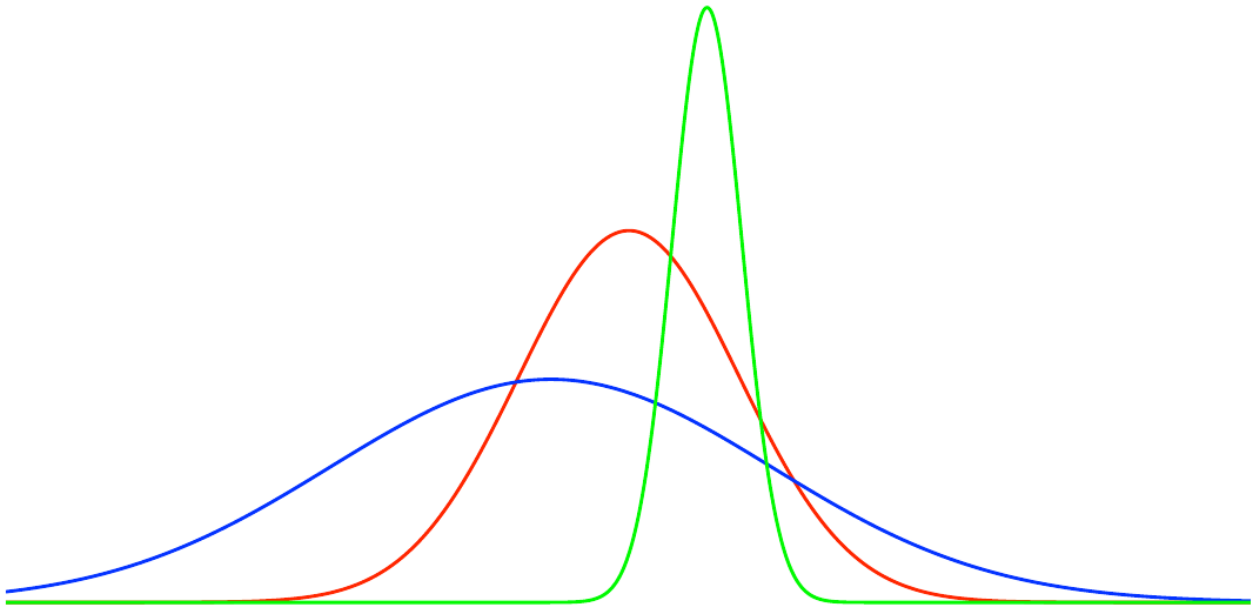
Marcello Restelli



- The more **uncertain** we are about an action–value
- The more important it is to **explore** that action
- It could turn out to be the **best action**

Optimism in Face of Uncertainty

Marcello Restelli



- The more **uncertain** we are about an action–value
- The more important it is to **explore** that action
- It could turn out to be the **best action**

Lower Bound

Marcello Restelli

- The performance of any algorithm is determined by **similarity** between optimal arm and other arms
- Hard problems have **similar–looking** arms with **different means**
- This is formally described by the gap Δ_a and the **similarity** in distributions $KL(R(\cdot|a)||R(\cdot,a^*))$

Theorem

*Lai and Robbins Asymptotic total regret is at least **logarithmic** in number of steps*

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

Lower Bound

- The performance of any algorithm is determined by **similarity** between optimal arm and other arms
- Hard problems have **similar–looking** arms with **different means**
- This is formally described by the gap Δ_a and the **similarity** in distributions $KL(R(\cdot|a)||R(\cdot,a^*))$

Theorem

Lai and Robbins Asymptotic total regret is at least ***logarithmic*** *in number of steps*

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

Lower Bound

- The performance of any algorithm is determined by **similarity** between optimal arm and other arms
- Hard problems have **similar–looking** arms with **different means**
- This is formally described by the gap Δ_a and the **similarity** in distributions $KL(R(\cdot|a)||R(\cdot,a^*))$

Theorem

Lai and Robbins Asymptotic total regret is at least ***logarithmic*** *in number of steps*

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

Lower Bound

- The performance of any algorithm is determined by **similarity** between optimal arm and other arms
- Hard problems have **similar–looking** arms with **different means**
- This is formally described by the gap Δ_a and the **similarity** in distributions $KL(R(\cdot|a)||R(\cdot,a^*))$

Theorem

Lai and Robbins Asymptotic total regret is at least ***logarithmic*** *in number of steps*

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

Upper Confidence Bounds

- Estimate an **upper confidence** $\hat{U}_t(a)$ for each action value
- Such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with **high probability**
- This depends on the **number of items** $N(a)$ has been selected
  - Small $N_t(a) \Rightarrow$ large $\hat{U}_t(a)$ (estimated value is **uncertain**)
  - Large $N_t(a) \Rightarrow$ small $\hat{U}_t(a)$ (estimated value is **accurate**)
- Select action maximizing **Upper Confidence Bound** (UCB)

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

Upper Confidence Bounds

- Estimate an **upper confidence** $\hat{U}_t(a)$ for each action value
- Such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with **high probability**
- This depends on the **number of items** $N(a)$ has been selected
  - Small $N_t(a) \Rightarrow$ large $\hat{U}_t(a)$ (estimated value is **uncertain**)
  - Large $N_t(a) \Rightarrow$ small $\hat{U}_t(a)$ (estimated value is **accurate**)
- Select action maximizing **Upper Confidence Bound** (UCB)

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

Upper Confidence Bounds

- Estimate an **upper confidence** $\hat{U}_t(a)$ for each action value
- Such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with **high probability**
- This depends on the **number of items** $N(a)$ has been selected
  - Small $N_t(a) \Rightarrow$ large $\hat{U}_t(a)$ (estimated value is **uncertain**)
  - Large $N_t(a) \Rightarrow$ small $\hat{U}_t(a)$ (estimated value is **accurate**)
- Select action maximizing **Upper Confidence Bound** (UCB)

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

Upper Confidence Bounds

- Estimate an **upper confidence** $\hat{U}_t(a)$ for each action value
- Such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with **high probability**
- This depends on the **number of items** $N(a)$ has been selected
  - Small $N_t(a) \Rightarrow$ large $\hat{U}_t(a)$ (estimated value is **uncertain**)
  - Large $N_t(a) \Rightarrow$ small $\hat{U}_t(a)$ (estimated value is **accurate**)
- Select action maximizing **Upper Confidence Bound** (UCB)

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

Upper Confidence Bounds

- Estimate an **upper confidence** $\hat{U}_t(a)$ for each action value
- Such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with **high probability**
- This depends on the **number of items** $N(a)$ has been selected
  - Small $N_t(a) \Rightarrow$ large $\hat{U}_t(a)$ (estimated value is **uncertain**)
  - Large $N_t(a) \Rightarrow$ small $\hat{U}_t(a)$ (estimated value is **accurate**)
- Select action maximizing **Upper Confidence Bound** (UCB)

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

Upper Confidence Bounds

Marcello Restelli

- Estimate an **upper confidence** $\hat{U}_t(a)$ for each action value
- Such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with **high probability**
- This depends on the **number of items** $N(a)$ has been selected
  - Small $N_t(a) \Rightarrow$ large $\hat{U}_t(a)$ (estimated value is **uncertain**)
  - Large $N_t(a) \Rightarrow$ small $\hat{U}_t(a)$ (estimated value is **accurate**)
- Select action maximizing **Upper Confidence Bound** (UCB)

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

Hoeffding's Inequality

Theorem

Hoeffding's Inequality Let $X_1, \ldots, X_N$ be i.i.d. random variables in $[0,1]$, and let $\overline{X}_N = \frac{1}{N}\sum_{i=1}^{N} X_i$ be the sample mean. Then

$$\mathbb{P}[\mathbb{E}[X] > \overline{X}_N + u] \leq e^{-2Nu^2}$$

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

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

Calculating Upper Confidence Bounds

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

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

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

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

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

Calculating Upper Confidence Bounds

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

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

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

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

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

Calculating Upper Confidence Bounds

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

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

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

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

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

Calculating Upper Confidence Bounds

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

$$\begin{aligned} e^{-2N_t(a)U_t(a)^2} &= p \\ U_t(a) &= \sqrt{\frac{-\log p}{2N_t(a)}} \end{aligned}$$

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

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

UCB1

Marcello Restelli

This leads to the UCB1 algorithm

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

Theorem

At time T, the expected total regret of the UCB policy is at most

$$\mathbb{E}[L_T] \leq 8 \log T \sum_{a | \Delta_a < \Delta_{a^*}} \frac{1}{\Delta_a} + \left(1 + \frac{\pi^2}{3}\right) \sum_{a \in \mathcal{A}} \Delta_a$$

Example: UCB vs ϵ–Greedy on 10–armed Bandit

Marcello Restelli

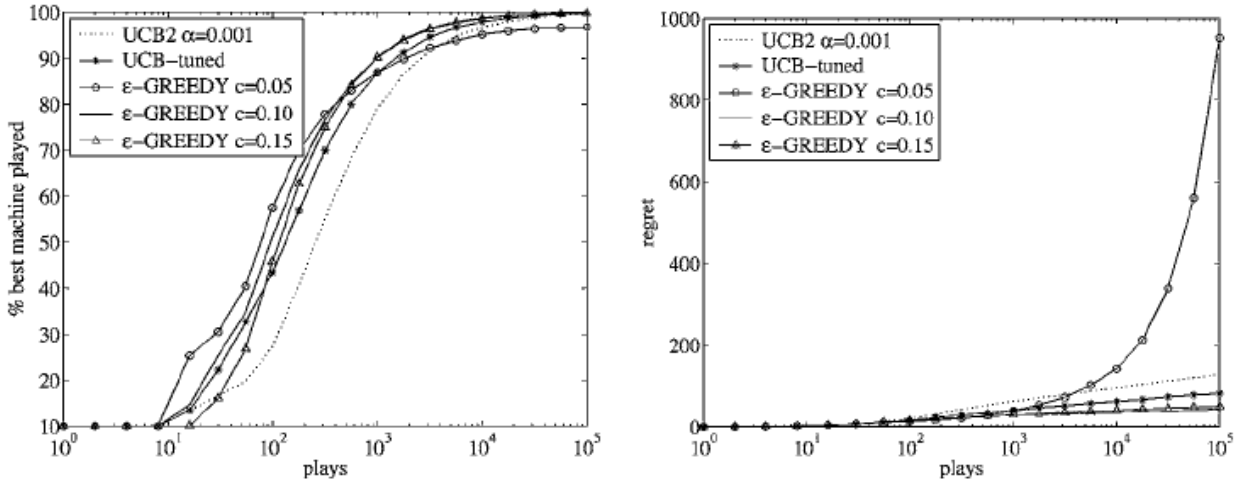


Figure 9. Comparison on distribution 11 (10 machines with parameters 0.9, 0.6, . . . , 0.6).

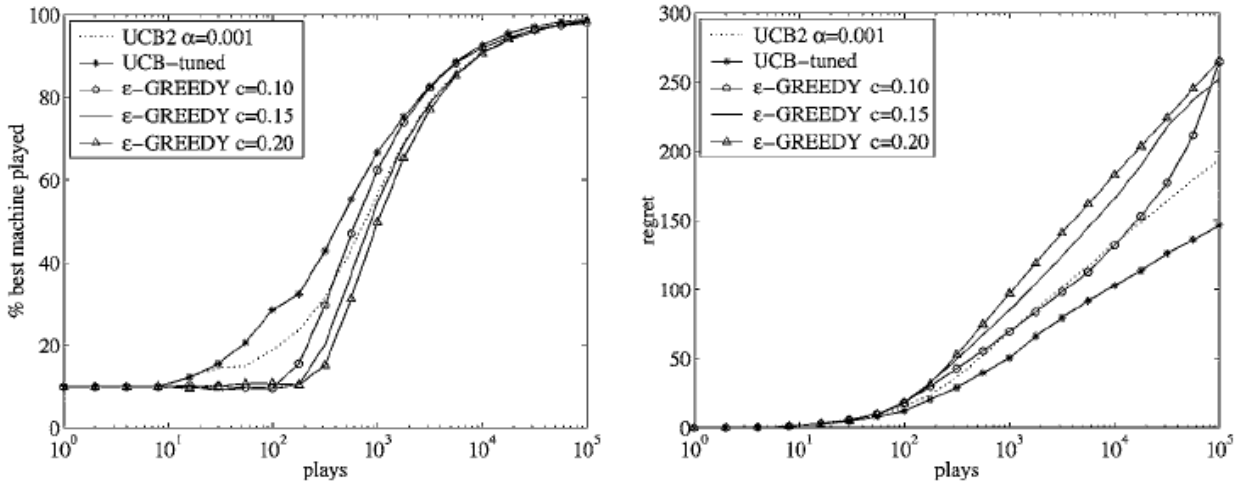


Figure 10. Comparison on distribution 12 (10 machines with parameters 0.9, 0.8, 0.8, 0.8, 0.7, 0.7, 0.7, 0.6, 0.6, 0.6).

Other Upper Confidence Bounds

Upper confidence bounds can be applied to other inequalities

- Bernstein's inequality
- Empirical Bernstein's inequality
- Chernoff inequality
- Azuma's inequality
- ...

The Adversarial Bandit Setting

- N arms
- At each round $t = 1, \ldots, T$
  - The learner chooses $I_t \in 1, \ldots, N$
  - At the same time the adversary selects reward vector $r_t = (r_{1,t}, \ldots, r_{N,t}) \in [0,1]^N$
  - The learner receives the reward $r_{I_t,t}$, while the rewards of the other arms are not received
- **Weak Regret** for algorithm A

$$L_T = G_{\max}(T) - \mathbb{E}\{G_A(T)\} = \max_j \sum_{t=1}^{T} R_t(a_j) - \sum_{t=1}^{T} R_t(A(t))$$

- In the stochastic setting there is no single best action that should be played all the time (like in the stochastic setting). *Weak* regret ignores this, but is simpler to analyze.

The Adversarial Bandit Setting

Marcello Restelli

- N arms
- At each round $t = 1, \ldots, T$
  - The learner chooses $I_t \in 1, \ldots, N$
  - At the same time the adversary selects reward vector $r_t = (r_{1,t}, \ldots, r_{N,t}) \in [0,1]^N$
  - The learner receives the reward $r_{I_t,t}$, while the rewards of the other arms are not received
- **Weak Regret** for algorithm A

$$L_T = G_{\max}(T) - \mathbb{E}\{G_A(T)\} = \max_j \sum_{t=1}^{T} R_t(a_j) - \sum_{t=1}^{T} R_t(A(t))$$

- In the stochastic setting there is no single best action that should be played all the time (like in the stochastic setting). *Weak* regret ignores this, but is simpler to analyze.

The Adversarial Bandit Setting

- N arms
- At each round $t = 1, \ldots, T$
  - The learner chooses $I_t \in 1, \ldots, N$
  - At the same time the adversary selects reward vector $r_t = (r_{1,t}, \ldots, r_{N,t}) \in [0,1]^N$
  - The learner receives the reward $r_{I_t,t}$, while the rewards of the other arms are not received
- **Weak Regret** for algorithm A

$$L_T = G_{\max}(T) - \mathbb{E}\{G_A(T)\} = \max_j \sum_{t=1}^{T} R_t(a_j) - \sum_{t=1}^{T} R_t(A(t))$$

- In the stochastic setting there is no single best action that should be played all the time (like in the stochastic setting). *Weak* regret ignores this, but is simpler to analyze.

The Adversarial Bandit Setting

- N arms
- At each round $t = 1, \ldots, T$
  - The learner chooses $I_t \in 1, \ldots, N$
  - At the same time the adversary selects reward vector $r_t = (r_{1,t}, \ldots, r_{N,t}) \in [0,1]^N$
  - The learner receives the reward $r_{I_t,t}$, while the rewards of the other arms are not received
- **Weak Regret** for algorithm A

$$L_T = G_{\max}(T) - \mathbb{E}\{G_A(T)\} = \max_j \sum_{t=1}^{T} R_t(a_j) - \sum_{t=1}^{T} R_t(A(t))$$

- In the stochastic setting there is no single best action that should be played all the time (like in the stochastic setting). *Weak* regret ignores this, but is simpler to analyze.

The Adversarial Bandit Setting

Marcello Restelli

- N arms
- At each round $t = 1, \ldots, T$
  - The learner chooses $I_t \in 1, \ldots, N$
  - At the same time the adversary selects reward vector $r_t = (r_{1,t}, \ldots, r_{N,t}) \in [0,1]^N$
  - The learner receives the reward $r_{I_t,t}$, while the rewards of the other arms are not received
- **Weak Regret** for algorithm A

$$L_T = G_{\max}(T) - \mathbb{E}\{G_A(T)\} = \max_j \sum_{t=1}^{T} R_t(a_j) - \sum_{t=1}^{T} R_t(A(t))$$

- In the stochastic setting there is no single best action that should be played all the time (like in the stochastic setting). *Weak* regret ignores this, but is simpler to analyze.

The Adversarial Bandit Setting

- N arms
- At each round $t = 1, \ldots, T$
  - The learner chooses $I_t \in 1, \ldots, N$
  - At the same time the adversary selects reward vector $r_t = (r_{1,t}, \ldots, r_{N,t}) \in [0,1]^N$
  - The learner receives the reward $r_{I_t,t}$, while the rewards of the other arms are not received
- **Weak Regret** for algorithm A

$$L_T = G_{\max}(T) - \mathbb{E}\{G_A(T)\} = \max_j \sum_{t=1}^{T} R_t(a_j) - \sum_{t=1}^{T} R_t(A(t))$$

- In the stochastic setting there is no single best action that should be played all the time (like in the stochastic setting). *Weak* regret ignores this, but is simpler to analyze.

The Adversarial Bandit Setting

- N arms
- At each round $t = 1, \dots, T$
  - The learner chooses $I_t \in 1, \dots, N$
  - At the same time the adversary selects reward vector $r_t = (r_{1,t}, \dots, r_{N,t}) \in [0,1]^N$
  - The learner receives the reward $r_{I_t,t}$, while the rewards of the other arms are not received
- **Weak Regret** for algorithm A

$$L_T = G_{\max}(T) - \mathbb{E}\{G_A(T)\} = \max_j \sum_{t=1}^{T} R_t(a_j) - \sum_{t=1}^{T} R_t(A(t))$$

- In the stochastic setting there is no single best action that should be played all the time (like in the stochastic setting). *Weak* regret ignores this, but is simpler to analyze.

Variation on Softmax
EXP3

- It is possible to drive regret down by **annealing** τ
- **EXP3**: Exponential weight algorithm for exploration and exploitation
- The probability of choosing arm a at time t is

$$\pi(a|t) = (1-\beta)\frac{w_t(a)}{\sum_{a' \in \mathcal{A}} w_t(a')} + \frac{\beta}{|\mathcal{A}|}$$

$$w_{t+1}(a) = \begin{cases} w_t(a) e^{\left(\eta \frac{r_t(a)}{\pi_t(a)}\right)} & \text{if arm } a \text{ is pulled at } t \\ w_t(a) & \text{otherwise} \end{cases}$$

- $\eta > 0$ (step size) and $\beta > 0$ (*egalitarianism factor*) are the parameters of the algorithm
- If $\beta = \eta = \sqrt{\frac{|\mathcal{A}| \log |\mathcal{A}|}{(e-1)T}}$ then **Regret**: $\mathbb{E}[L_T] \leq O(\sqrt{T|\mathcal{A}| \log |\mathcal{A}|})$

Variation on Softmax
EXP3

- It is possible to drive regret down by **annealing** τ
- **EXP3**: Exponential weight algorithm for exploration and exploitation
- The probability of choosing arm a at time t is

$$\pi(a|t) = (1-\beta)\frac{w_t(a)}{\sum_{a' \in \mathcal{A}} w_t(a')} + \frac{\beta}{|\mathcal{A}|}$$

$$w_{t+1}(a) = \begin{cases} w_t(a) e^{\left(\eta \frac{r_t(a)}{\pi_t(a)}\right)} & \text{if arm } a \text{ is pulled at } t \\ w_t(a) & \text{otherwise} \end{cases}$$

- $\eta > 0$ (step size) and $\beta > 0$ (*egalitarianism factor*) are the parameters of the algorithm
- If $\beta = \eta = \sqrt{\frac{|\mathcal{A}| \log |\mathcal{A}|}{(e-1)T}}$ then **Regret**:
$\mathbb{E}[L_T] \leq O(\sqrt{T|\mathcal{A}| \log |\mathcal{A}|})$

Variation on Softmax
EXP3

- It is possible to drive regret down by **annealing** τ
- **EXP3**: Exponential weight algorithm for exploration and exploitation
- The probability of choosing arm a at time t is

$$\pi(a|t) = (1-\beta)\frac{w_t(a)}{\sum_{a' \in \mathcal{A}} w_t(a')} + \frac{\beta}{|\mathcal{A}|}$$

$$w_{t+1}(a) = \begin{cases} w_t(a) e^{\left(\eta \frac{r_t(a)}{\pi_t(a)}\right)} & \text{if arm } a \text{ is pulled at } t \\ w_t(a) & \text{otherwise} \end{cases}$$

- $\eta > 0$ (step size) and $\beta > 0$ (*egalitarianism factor*) are the parameters of the algorithm
- If $\beta = \eta = \sqrt{\frac{|\mathcal{A}| \log |\mathcal{A}|}{(e-1)T}}$ then **Regret**: $\mathbb{E}[L_T] \leq O(\sqrt{T|\mathcal{A}| \log |\mathcal{A}|})$

Variation on Softmax
EXP3

- It is possible to drive regret down by **annealing** τ
- **EXP3**: Exponential weight algorithm for exploration and exploitation
- The probability of choosing arm a at time t is

$$\pi(a|t) = (1-\beta)\frac{w_t(a)}{\sum_{a' \in \mathcal{A}} w_t(a')} + \frac{\beta}{|\mathcal{A}|}$$

$$w_{t+1}(a) = \begin{cases} w_t(a) e^{\left(\eta \frac{r_t(a)}{\pi_t(a)}\right)} & \text{if arm } a \text{ is pulled at } t \\ w_t(a) & \text{otherwise} \end{cases}$$

- $\eta > 0$ (step size) and $\beta > 0$ (*egalitarianism factor*) are the parameters of the algorithm
- If $\beta = \eta = \sqrt{\frac{|\mathcal{A}| \log |\mathcal{A}|}{(e-1)T}}$ then **Regret**: $\mathbb{E}[L_T] \leq O(\sqrt{T|\mathcal{A}| \log |\mathcal{A}|})$

Variation on Softmax
EXP3

- It is possible to drive regret down by **annealing** τ
- **EXP3**: Exponential weight algorithm for exploration and exploitation
- The probability of choosing arm a at time t is

$$\pi(a|t) = (1-\beta)\frac{w_t(a)}{\sum_{a' \in \mathcal{A}} w_t(a')} + \frac{\beta}{|\mathcal{A}|}$$

$$w_{t+1}(a) = \begin{cases} w_t(a) e^{\left(\eta \frac{r_t(a)}{\pi_t(a)}\right)} & \text{if arm } a \text{ is pulled at } t \\ w_t(a) & \text{otherwise} \end{cases}$$

- $\eta > 0$ (step size) and $\beta > 0$ (*egalitarianism factor*) are the parameters of the algorithm
- If $\beta = \eta = \sqrt{\frac{|\mathcal{A}| \log |\mathcal{A}|}{(e-1)T}}$ then **Regret**: $\mathbb{E}[L_T] \leq O(\sqrt{T|\mathcal{A}| \log |\mathcal{A}|})$

Outline

Bayes' Theorem

- X is an hidden variable
- Z is an observed variable
- We update our guess on X given the value of Z we see
- The posterior becomes the new prior

Bayes' Theorem

- X is an hidden variable
- Z is an observed variable
- We update our guess on X given the value of Z we see
- The posterior becomes the new prior

Bayes' Theorem

- X is an hidden variable
- Z is an observed variable
- We update our guess on X given the value of Z we see

- The posterior becomes the new prior

Bayes' Theorem

- X is an hidden variable
- Z is an observed variable
- We update our guess on X given the value of Z we see

$$\underbrace{\mathbb{P}(X \mid Z)}_{\text{posterior}} = \frac{\overbrace{\mathbb{P}(X)}^{\text{prior}} \overbrace{\mathbb{P}(Z \mid X)}^{\textit{likelihood}}}{P(Z)}$$

- The posterior becomes the new prior

Bayes' Theorem

- X is an hidden variable
- Z is an observed variable
- We update our guess on X given the value of Z we see

$$\underbrace{\mathbb{P}(X \mid Z)}_{\text{posterior}} \propto \overbrace{\mathbb{P}(X)}^{\text{prior}} \overbrace{\mathbb{P}(Z \mid X)}^{\textit{likelihood}}$$

- The posterior becomes the new prior

Bayes' Theorem

Marcello Restelli

- X is an hidden variable
- Z is an observed variable
- We update our guess on X given the value of Z we see

$$\underbrace{\mathbb{P}(X \mid Z)}_{\text{posterior}} \propto \overbrace{\mathbb{P}(X)}^{\text{prior}} \overbrace{\mathbb{P}(Z \mid X)}^{\textit{likelihood}}$$

- The posterior becomes the new prior

Bayesian Bandit
The Bernoulli case

- We assume reward is either 0 or 1 (observation)
- We model rewards as **Bernoulli** variables: $R(a_j) \sim Ber(p_j)$, where p_j is **unknown** (hidden)

$$\underbrace{\mathbb{P}(r \mid p_j)}_{\text{likelihood}} = p_j^r (1-p_j)^{(1-r)}$$

- We model belief with **Beta** distributions: $p \sim Beta(\alpha_j, \beta_j)$

$$\underbrace{\mathbb{P}(p)}_{\text{prior}} = \frac{p_j^{\alpha_j - 1}(1-p_j)^{\beta_j - 1}}{B(\alpha_j, \beta_j)}$$

$B(\alpha_j, \beta_j) = \frac{\Gamma(\alpha_j)\Gamma(\beta_j)}{\Gamma(\alpha_j+\beta_j)}$ is just a normalization term

Bayesian Bandit
The Bernoulli case

- We assume reward is either 0 or 1 (observation)
- We model rewards as **Bernoulli** variables: $R(a_j) \sim Ber(p_j)$, where p_j is **unknown** (hidden)

$$\underbrace{\mathbb{P}(r \mid p_j)}_{\text{likelihood}} = p_j^r (1 - p_j)^{(1-r)}$$

- We model belief with **Beta** distributions: $p \sim Beta(\alpha_j, \beta_j)$

$$\underbrace{\mathbb{P}(p)}_{\text{prior}} = \frac{p_j^{\alpha_j - 1}(1 - p_j)^{\beta_j - 1}}{B(\alpha_j, \beta_j)}$$

$B(\alpha_j, \beta_j) = \frac{\Gamma(\alpha_j)\Gamma(\beta_j)}{\Gamma(\alpha_j + \beta_j)}$ is just a normalization term

Bayesian Bandit
The Bernoulli case

Marcello Restelli

- We assume reward is either 0 or 1 (observation)
- We model rewards as **Bernoulli** variables: $R(a_j) \sim Ber(p_j)$, where p_j is **unknown** (hidden)

$$\underbrace{\mathbb{P}(r \mid p_j)}_{\text{likelihood}} = p_j^r (1 - p_j)^{(1-r)}$$

- We model belief with **Beta** distributions: $p \sim Beta(\alpha_j, \beta_j)$

$$\underbrace{\mathbb{P}(p)}_{\text{prior}} = \frac{p_j^{\alpha_j - 1}(1 - p_j)^{\beta_j - 1}}{B(\alpha_j, \beta_j)}$$

$B(\alpha_j, \beta_j) = \frac{\Gamma(\alpha_j)\Gamma(\beta_j)}{\Gamma(\alpha_j + \beta_j)}$ is just a normalization term

Marcello Restelli

- The Beta distribution is a **conjugate prior** for Bernoulli variables: a Beta times a Bernoulli is a Beta:

$$\underbrace{\mathbb{P}(p \mid r)}_{\text{posterior}} = \mathit{Beta}(\alpha_j + r, \beta_j + 1 - r)$$

- We can make choices based on our current belief, and update belief based on observations
- The more data we observe, the more the Betas become peaked
- *Visualize Beta distributions:* `http://www.distributome.org/js/sim/BetaSimulation.html`

Bayesian Bandit
Updating belief

- The Beta distribution is a **conjugate prior** for Bernoulli variables: a Beta times a Bernoulli is a Beta:

$$\underbrace{\mathbb{P}(p \mid r)}_{\text{posterior}} = \mathit{Beta}(\alpha_j + r, \beta_j + 1 - r)$$

- We can make choices based on our current belief, and update belief based on observations
- The more data we observe, the more the Betas become peaked
- *Visualize Beta distributions:* `http://www.distributome.org/js/sim/BetaSimulation.html`

Bayesian Bandit
Updating belief

- The Beta distribution is a **conjugate prior** for Bernoulli variables: a Beta times a Bernoulli is a Beta:

$$\underbrace{\mathbb{P}(p \mid r)}_{\text{posterior}} = \mathit{Beta}(\alpha_j + r, \beta_j + 1 - r)$$

- We can make choices based on our current belief, and update belief based on observations
- The more data we observe, the more the Betas become peaked
- *Visualize Beta distributions:* `http://www.distributome.org/js/sim/BetaSimulation.html`

Bayesian Bandit
Updating belief

Marcello Restelli

- The Beta distribution is a **conjugate prior** for Bernoulli variables: a Beta times a Bernoulli is a Beta:

$$\underbrace{\mathbb{P}(p \mid r)}_{\text{posterior}} = \mathit{Beta}(\alpha_j + r, \beta_j + 1 - r)$$

- We can make choices based on our current belief, and update belief based on observations
- The more data we observe, the more the Betas become peaked
- *Visualize Beta distributions:* `http://www.distributome.org/js/sim/BetaSimulation.html`

Thompson Sampling

Marcello Restelli

Initialize $\alpha_j = \beta_j = 1$ for $j = 1, \ldots, N$
loop
  for $j = 1, \ldots, N$ **do**
    sample $\widehat{p}_j \sim Beta(\alpha_j, \beta_j)$
  end for
  Select arm $k = \arg\max_j \widehat{p}_j$
  Observe $r \sim R(a_k)$
  $\alpha_k \leftarrow \alpha_k + r$
  $\beta_k \leftarrow \beta_k + 1 - r$
end loop

Outline

MAB with Infinitely Many Arms

Marcello Restelli

- **Unstructured** set of actions
  - UCB Arm-Increasing Rule
- **Structured** set of actions
  - Linear Bandits
  - Lipschitz Bandits
  - Unimodal
  - Bandits in trees
- **Context**
  - Contextual bandit: transition between different MABs (states), but independently from the chosen arm (**no dynamics**)

MAB with Infinitely Many Arms

- **Unstructured** set of actions
  - UCB Arm-Increasing Rule
- **Structured** set of actions
  - Linear Bandits
  - Lipschitz Bandits
  - Unimodal
  - Bandits in trees
- **Context**
  - Contextual bandit: transition between different MABs (states), but independently from the chosen arm (**no dynamics**)

MAB with Infinitely Many Arms

Marcello Restelli

- **Unstructured** set of actions
  - UCB Arm-Increasing Rule
- **Structured** set of actions
  - Linear Bandits
  - Lipschitz Bandits
  - Unimodal
  - Bandits in trees
- **Context**
  - Contextual bandit: transition between different MABs (states), but independently from the chosen arm (**no dynamics**)

MAB with Infinitely Many Arms

Marcello Restelli

- **Unstructured** set of actions
  - UCB Arm-Increasing Rule
- **Structured** set of actions
  - Linear Bandits
  - Lipschitz Bandits
  - Unimodal
  - Bandits in trees
- **Context**
  - Contextual bandit: transition between different MABs (states), but independently from the chosen arm (**no dynamics**)

MAB with Infinitely Many Arms

Marcello Restelli

- **Unstructured** set of actions
  - UCB Arm-Increasing Rule
- **Structured** set of actions
  - Linear Bandits
  - Lipschitz Bandits
  - Unimodal
  - Bandits in trees
- **Context**
  - Contextual bandit: transition between different MABs (states), but independently from the chosen arm (**no dynamics**)

MAB with Infinitely Many Arms

Marcello Restelli

- **Unstructured** set of actions
  - UCB Arm-Increasing Rule
- **Structured** set of actions
  - Linear Bandits
  - Lipschitz Bandits
  - Unimodal
  - Bandits in trees
- **Context**
  - Contextual bandit: transition between different MABs (states), but independently from the chosen arm (**no dynamics**)

MAB with Infinitely Many Arms

Marcello Restelli

- **Unstructured** set of actions
  - UCB Arm-Increasing Rule
- **Structured** set of actions
  - Linear Bandits
  - Lipschitz Bandits
  - Unimodal
  - Bandits in trees
- **Context**
  - Contextual bandit: transition between different MABs (states), but independently from the chosen arm (**no dynamics**)

MAB with Infinitely Many Arms

Marcello Restelli

- **Unstructured** set of actions
  - UCB Arm-Increasing Rule
- **Structured** set of actions
  - Linear Bandits
  - Lipschitz Bandits
  - Unimodal
  - Bandits in trees
- **Context**
  - Contextual bandit: transition between different MABs (states), but independently from the chosen arm (**no dynamics**)

MAB with Infinitely Many Arms

- **Unstructured** set of actions
  - UCB Arm-Increasing Rule
- **Structured** set of actions
  - Linear Bandits
  - Lipschitz Bandits
  - Unimodal
  - Bandits in trees
- **Context**
  - Contextual bandit: transition between different MABs (states), but independently from the chosen arm (**no dynamics**)