Probability

9.8 Multi-Armed bandits

Aleks Jovcic
Slides by Alex Tsun

Agenda

- The Multi-Armed Bandit (MAB) Problem
- Greedy/Epsilon-Greedy
- Upper Confidence Bound (UCB)
- Thompson Sampling
- Modern Hypothesis Testing

Multi-Armed Bandit (MAB) Problem

- K Slot Machines {1,2,...,K} (aka "Bandits" with "Arms").

Multi-Armed Bandit (MAB) Problem

- K Slot Machines {1,2,...,K} (aka "Bandits" with "Arms").
- At each time step t=1,2,...,T: Pull an arm $a_t \in \{1,2,...,K\}$ and observe random reward (each arm is independent, and has some reward distribution which doesn't change over time).

Multi-Armed Bandit (MAB) Problem

- K Slot Machines {1,2,...,K} (aka "Bandits" with "Arms").
- At each time step t=1,2,...,T: Pull an arm $a_t \in \{1,2,...,K\}$ and observe random reward (each arm is independent, and has some reward distribution which doesn't change over time).
- <u>**Goal**</u>: Maximize total (expected) reward after T time steps.

Multi-Armed Bandit (MAB) Problem

- K Slot Machines {1,2,...,K} (aka "Bandits" with "Arms").
- At each time step t=1,2,...,T: Pull an arm $a_t \in \{1,2,\ldots,K\}$ and observe random reward (each arm is independent, and has some reward distribution which doesn't change over time).
- **Goal**: Maximize total (expected) reward after T time steps.

- **Problem**: At each time step, decide which arm to pull based on past history of rewards.

Multi-Armed Bandit (MAB) Problem

Below has the reward distribution of each of the K=3 arms.

What's your strategy to maximize your total (expected) reward?

$Poi(\lambda = 1.36)$

$Bin(n = 10, p = 0.4)$

$\mathcal{N}(\mu = -1, \sigma^2 = 4)$

Multi-Armed Bandit (MAB) Problem

Below has the reward distribution of each of the K=3 arms.

What's your strategy to maximize your total (expected) reward?

$Poi(\lambda = 1.36)$

$Bin(n = 10, p = 0.4)$

$\mathcal{N}(\mu = -1, \sigma^2 = 4)$

Pull arm 2 every time since it has the highest expected reward!

Multi-Armed Bandit (MAB) Problem

Well actually, we don't know the reward distributions :(.

Multi-Armed Bandit (MAB) Problem

Well actually, we don't know the reward distributions :(.

Have to **estimate** all K expectations

Multi-Armed Bandit (MAB) Problem

Well actually, we don't know the reward distributions :(.

Have to **estimate** all K expectations, WHILE simultaneously maximizing reward!

Multi-Armed Bandit (MAB) Problem

Well actually, we don't know the reward distributions :(.

Have to **estimate** all K expectations, WHILE simultaneously maximizing reward!

This is a hard problem - we know nothing about the K reward distributions!

Multi-Armed Bandit (MAB) Problem

Need to balance the tradeoff between:

Exploitation: Pulling arm(s) we know to be "good".
Exploration: Pulling other arms in the hopes they are also "good" or even better.

Bernoulli Bandits

We will handle the case of Bernoulli-bandits. That is, reward of arm a $\in$ {1,2,…,K} is Ber(p_a) .

$Ber(p_1)$

$Ber(p_2)$

$Ber(p_3)$

Bernoulli Bandits

We will handle the case of Bernoulli-bandits. That is, reward of arm a $\in$ {1,2,...,K} is Ber(p_a) .

We don't know these!

Bernoulli Bandits

We will handle the case of Bernoulli-bandits. That is, reward of arm a $\in$ {1,2,...,K} is Ber(p_a) .

Observe: The expected reward of arm a is just p_a (expectation of Bernoulli).

$Ber(p_1)$

$Ber(p_2)$

$Ber(p_3)$

We don't know these!

Regret

Regret is the difference between:

- The best possible expected reward (always pull the best arm)
- The actual reward you got

Regret

Regret is the difference between:

- The best possible expected reward (always pull the best arm)
- The actual reward you got

Let $p^* = \max_{i \in \{1,2,\ldots,K\}} p_i$ denote the highest expected reward from one of the K arms.

Regret

Regret is the difference between:

- The best possible expected reward (always pull the best arm)
- The actual reward you got

Let $p^* = \max_{i \in \{1,2,\ldots,K\}} p_i$ denote the highest expected reward from one of the K arms.

$$\text{Regret}(T) = Tp^* - \text{Reward}(T)$$

At some "time" T (after T arm pulls)

Regret

Regret is the difference between:

- The best possible expected reward (always pull the best arm)
- The actual reward you got

Let $p^* = \max_{i \in \{1,2,\dots,K\}} p_i$ denote the highest expected reward from one of the K arms.

$$\text{Regret}(T) = Tp^* - \text{Reward}(T)$$

At some "time" T (after T arm pulls)

Regret

Regret is the difference between:

- The best possible expected reward (always pull the best arm)
- The actual reward you got

Let $p^* = \max_{i \in \{1,2,\ldots,K\}} p_i$ denote the highest expected reward from one of the K arms.

$$\text{Regret}(T) = Tp^* - \text{Reward}(T)$$

At some "time" T (after T arm pulls)

Regret

Regret is the difference between:

- The best possible expected reward (always pull the best arm)
- The actual reward you got

Let $p^* = \max_{i \in \{1,2,\ldots,K\}} p_i$ denote the highest expected reward from one of the K arms.

$$\text{Regret}(T) = Tp^* - \text{Reward}(T)$$

$$\text{Avg-Regret}(T) = p^* - \frac{\text{Reward}(T)}{T}$$

Regret

Regret is the difference between:

- The best possible expected reward (always pull the best arm)
- The actual reward you got

Let $p^* = \max_{i \in \{1,2,\ldots,K\}} p_i$ denote the highest expected reward from one of the K arms.

$$\text{Regret}(T) = Tp^* - \text{Reward}(T)$$

$$\text{Avg-Regret}(T) = p^* - \frac{\text{Reward}(T)}{T}$$

Want Avg-Regret(T) → 0 as T → ∞ . **Minimizing Regret = Maximizing Reward.**

(Bernoulli) Bandit Framework

How do we choose an arm at each time step (depending on past history), to maximize our total reward?

Algorithm 1 (Bernoulli) Bandit Framework

1: Have K arms, where pulling arm $i \in \{1, \ldots, K\}$ gives $Ber(p_i)$ reward ▷ p_i's all unknown.

(Bernoulli) Bandit Framework

How do we choose an arm at each time step (depending on past history), to maximize our total reward?

Algorithm 1 (Bernoulli) Bandit Framework

Have K arms, where pulling arm $i \in \{1, \ldots, K\}$ gives $Ber(p_i)$ reward ▷ p_i's all unknown.
for $t = 1, \ldots, T$ **do**
    At time t, pull arm $a_t \in \{1, \ldots, K\}$. ▷ How do we do decide which arm?

(Bernoulli) Bandit Framework

How do we choose an arm at each time step (depending on past history), to maximize our total reward?

Algorithm 1 (Bernoulli) Bandit Framework

Have K arms, where pulling arm $i \in \{1, \ldots, K\}$ gives $Ber(p_i)$ reward ▷ p_i's all unknown.
for $t = 1, \ldots, T$ **do**
    At time t, pull arm $a_t \in \{1, \ldots, K\}$. ▷ How do we do decide which arm?
    Receive reward $r_t \sim Ber(p_{a_t})$. ▷ Reward is either 1 or 0.

(Bernoulli) Bandit Framework

How do we choose an arm at each time step (depending on past history), to maximize our total reward?

Algorithm 1 (Bernoulli) Bandit Framework

Have K arms, where pulling arm $i \in \{1, \ldots, K\}$ gives $Ber(p_i)$ reward ▷ p_i's all unknown.
for $t = 1, \ldots, T$ **do**
    At time t, pull arm $a_t \in \{1, \ldots, K\}$. ▷ How do we do decide which arm?
    Receive reward $r_t \sim Ber(p_{a_t})$. ▷ Reward is either 1 or 0.

This is the focus of the rest of this lecture!

Motivation: Clinical Trials

K = 4 Arms (Treatments)

Motivation: Clinical Trials

$K = 4$ Arms (Treatments)

For patient t, prescribe treatment $a_t \in \{1,2,3,4\}$.

Motivation: Clinical Trials

K = 4 Arms (Treatments)

For patient t, prescribe treatment $a_t \in \{1,2,3,4\}$.

Observe reward $r_t \in \{0, 1\}$. (1 if healed, 0 if not)

Motivation: Clinical Trials

K = 4 Arms (Treatments)

For patient t, prescribe treatment $a_t \in \{1,2,3,4\}$.

Observe reward $r_t \in \{0, 1\}$. (1 if healed, 0 if not)

Maximize: Total number of patients healed.

Motivation: Recommending Movies

K Movies

For visitor t, recommend movie $a_t \in \{1,2,\ldots,K\}$.

Motivation: Recommending Movies

K Movies

For visitor t, recommend movie $a_t \in \{1,2,\ldots,K\}$.

Observe reward $r_t \in \{1,2,3,4,5\}$. (rating)

Maximize: Total/average rating of recommendations.

Motivation: Real Life?? (Food)

K Cuisines/Dishes (a ton)

For meal t, eat dish $a_t \in \{1,2,\ldots,K\}$.

Motivation: Real Life?? (Food)

K Cuisines/Dishes (a ton)

For meal t, eat dish $a_t \in \{1,2,\ldots,K\}$.

Observe reward $r_t \in \{1,2,3,4,5\}$. (happiness rating)

Maximize: Total/average happiness :)

Motivation: Real Life?? (Food)

K Cuisines/Dishes (a ton)

For meal t, eat dish $a_t \in \{1,2,\ldots,K\}$.

Observe reward $r_t \in \{1,2,3,4,5\}$. (happiness rating)

Maximize: Total/average happiness :)

The Question of the Day: Explore or Exploit????

Motivation: Real Life?? (Activities)

K Activities

On day t, do activity $a_t \in \{1,2,\ldots,K\}$.

Observe reward $r_t \in \{1,2,3,4,5\}$. (happiness rating)

Maximize: Total/average happiness :)

The Question of the Day: Explore or Exploit????

Any Ideas on what strategy we can use???

Greedy (Naive) Algorithm

Algorithm 2 Greedy (Naive) Strategy for Bernoulli Bandits

1: Choose a number of times M to pull each arm initially, with $KM \leq T$.

Greedy (Naive) Algorithm

Algorithm 2 Greedy (Naive) Strategy for Bernoulli Bandits

Choose a number of times M to pull each arm initially, with $KM \leq T$.
for $i = 1, 2, \ldots, K$ **do**
    Pull arm i M times, observing iid rewards $r_{i1}, \ldots, r_{iM} \sim Ber(p_i)$.
    Estimate $\hat{p}_i = \dfrac{\sum_{j=1}^{M} r_{ij}}{M}$.

Greedy (Naive) Algorithm

Algorithm 2 Greedy (Naive) Strategy for Bernoulli Bandits

Choose a number of times M to pull each arm initially, with $KM \leq T$.
for $i = 1, 2, \ldots, K$ **do**
    Pull arm i M times, observing iid rewards $r_{i1}, \ldots, r_{iM} \sim Ber(p_i)$.
    Estimate $\hat{p}_i = \frac{\sum_{j=1}^{M} r_{ij}}{M}$.
Determine best (empirical) arm $a^* = \arg\max_{i \in \{1,2,\ldots,K\}} \hat{p}_i$.   ▷ We could be wrong...

Greedy (Naive) Algorithm

Algorithm 2 Greedy (Naive) Strategy for Bernoulli Bandits

Choose a number of times M to pull each arm initially, with $KM \leq T$.
for $i = 1, 2, \ldots, K$ **do**
    Pull arm i M times, observing iid rewards $r_{i1}, \ldots, r_{iM} \sim Ber(p_i)$.
    Estimate $\hat{p}_i = \frac{\sum_{j=1}^{M} r_{ij}}{M}$.
Determine best (empirical) arm $a^* = \arg\max_{i \in \{1,2,\ldots,K\}} \hat{p}_i$. ▷ We could be wrong...
for $t = KM + 1, KM + 2, \ldots, T$ **do**:
    Pull arm $a_t = a^*$. ▷ Pull the same arm for the rest of time.
    Receive reward $r_t \sim Ber(p_{a_t})$.

Greedy (Naive) Algorithm

Algorithm 2 Greedy (Naive) Strategy for Bernoulli Bandits

```
Choose a number of times M to pull each arm initially, with KM ≤ T.
for i = 1, 2, ..., K do
    Pull arm i M times, observing iid rewards r_{i1}, ..., r_{iM} ~ Ber(p_i).
    Estimate p̂_i = (Σ_{j=1}^{M} r_{ij}) / M.
Determine best (empirical) arm a* = argmax_{i∈{1,2,...,K}} p̂_i.    ▹ We could be wrong...
for t = KM + 1, KM + 2, ..., T do:
    Pull arm a_t = a*.                         ▹ Pull the same arm for the rest of time.
    Receive reward r_t ~ Ber(p_{a_t}).
```

If we make a mistake, we will regret our decision for the rest of time....

Can we not do all of our exploration at the beginning?

Epsilon-Greedy Algorithm

ε

Explore with probability epsilon!

Algorithm 3 ε-Greedy Strategy for Bernoulli Bandits

Choose a number of times M to pull each arm initially, with $KM \leq T$.
for $i = 1, 2, \ldots, K$ **do**
    Pull arm i M times, observing iid rewards $r_{i1}, \ldots, r_{iM} \sim Ber(p_i)$.
    Estimate $\hat{p}_i = \frac{\sum_{j=1}^{M} r_{ij}}{M}$.

Epsilon-Greedy Algorithm

ε

Explore with probability epsilon!

Algorithm 3 ε-Greedy Strategy for Bernoulli Bandits

Choose a number of times M to pull each arm initially, with $KM \leq T$.
for $i = 1, 2, \ldots, K$ **do**
    Pull arm i M times, observing iid rewards $r_{i1}, \ldots, r_{iM} \sim Ber(p_i)$.
    Estimate $\hat{p}_i = \dfrac{\sum_{j=1}^{M} r_{ij}}{M}$.
for $t = KM + 1, KM + 2, \ldots, T$ **do**:
    if $Ber(\varepsilon) == 1$: **then** ▹ With probability ε, explore.
        Pull arm $a_t = Unif(1, K)$ (discrete). ▹ Choose a uniformly random arm.

Epsilon-Greedy Algorithm

ε

Explore with probability epsilon!

Algorithm 3 ε-Greedy Strategy for Bernoulli Bandits

```
Choose a number of times M to pull each arm initially, with KM ≤ T.
for i = 1, 2, ..., K do
    Pull arm i M times, observing iid rewards r_{i1}, ..., r_{iM} ~ Ber(p_i).
    Estimate p̂_i = (Σ_{j=1}^{M} r_{ij}) / M.
for t = KM + 1, KM + 2, ..., T do:
    if Ber(ε) == 1: then                          ▷ With probability ε, explore.
        Pull arm a_t = Unif(1, K) (discrete).     ▷ Choose a uniformly random arm.
    else                                          ▷ With probability 1 − ε, exploit.
        Pull arm a_t = arg max_{i∈{1,2,...,K}} p̂_i.  ▷ Choose arm with highest estimated reward.
```

Epsilon-Greedy Algorithm

ε

Explore with probability epsilon!

Algorithm 3 ε-Greedy Strategy for Bernoulli Bandits

Choose a number of times M to pull each arm initially, with $KM \leq T$.
for $i = 1, 2, \ldots, K$ **do**
    Pull arm i M times, observing iid rewards $r_{i1}, \ldots, r_{iM} \sim Ber(p_i)$.
    Estimate $\hat{p}_i = \frac{\sum_{j=1}^{M} r_{ij}}{M}$.
for $t = KM + 1, KM + 2, \ldots, T$ **do:**
    if $Ber(\varepsilon) == 1$: **then** ▷ With probability ε, explore.
        Pull arm $a_t = Unif(1, K)$ (discrete). ▷ Choose a uniformly random arm.
    else ▷ With probability $1 - \varepsilon$, exploit.
        Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \hat{p}_i$. ▷ Choose arm with highest estimated reward.
    Receive reward $r_t \sim Ber(p_{a_t})$.
    Update $\hat{p}_{a_t}$ (using newly observed reward r_t).

Epsilon-Greedy Algorithm

ε 

Explore with probability epsilon!

Algorithm 3 ε-Greedy Strategy for Bernoulli Bandits

Choose a number of times M to pull each arm initially, with $KM \leq T$.
for $i = 1, 2, \ldots, K$ **do**
  Pull arm i M times, observing iid rewards $r_{i1}, \ldots, r_{iM} \sim Ber(p_i)$.
  Estimate $\hat{p}_i = \frac{\sum_{j=1}^{M} r_{ij}}{M}$.
for $t = KM + 1, KM + 2, \ldots, T$ **do:**
  if $Ber(\varepsilon) == 1$: **then** ▷ With probability ε, explore.
    Pull arm $a_t = Unif(1, K)$ (discrete). ▷ Choose a uniformly random arm.
  else ▷ With probability $1 - \varepsilon$, exploit.
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \hat{p}_i$. ▷ Choose arm with highest estimated reward.
  Receive reward $r_t \sim Ber(p_{a_t})$.
  Update $\hat{p}_{a_t}$ (using newly observed reward r_t).

Can we explore more “naturally”?

Upper Confidence Bound (UCB) Algorithm

This algorithm constructs confidence intervals for the estimates of each arm, and chooses the arm with the highest **upper** confidence bound (if the confidence interval is [a,b], we compare only the value of b)

Upper Confidence Bound (UCB) Algorithm

This algorithm constructs confidence intervals for the estimates of each arm, and chooses the arm with the highest **upper** confidence bound (if the confidence interval is [a,b], we compare only the value of b)

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
    Pull arm i once, observing $r_i \sim Ber(p_i)$.
    Estimate $\hat{p}_i = r_i/1$.   ▷ Each estimate $\hat{p}_i$ will initially either be 1 or 0.

Upper Confidence Bound (UCB) Algorithm

This algorithm constructs confidence intervals for the estimates of each arm, and chooses the arm with the highest **upper** confidence bound (if the confidence interval is [a,b], we compare only the value of b)

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
    Pull arm i once, observing $r_i \sim Ber(p_i)$.
    Estimate $\hat{p}_i = r_i/1$.   ▷ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \ldots, T$ **do**:
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}} \right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.

Upper Confidence Bound (UCB) Algorithm

This algorithm constructs confidence intervals for the estimates of each arm, and chooses the arm with the highest **upper** confidence bound (if the confidence interval is [a,b], we compare only the value of b)

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

---

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

  Pull arm i once, observing $r_i \sim Ber(p_i)$.

  Estimate $\hat{p}_i = r_i / 1$. ▹ Each estimate $\hat{p}_i$ will initially either be 1 or 0.

for $t = K+1, K+2, \ldots, T$ **do**:

  Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}} \right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.

Point estimate/
Max-likelihood estimate

Upper Confidence Bound (UCB) Algorithm

This algorithm constructs confidence intervals for the estimates of each arm, and chooses the arm with the highest **upper** confidence bound (if the confidence interval is [a,b], we compare only the value of b)

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
    Pull arm i once, observing $r_i \sim Ber(p_i)$.
    Estimate $\hat{p}_i = r_i/1$.   ▷ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \ldots, T$ **do**:
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}} \right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.

Point estimate/
Max-likelihood estimate

Takes the upper part of of
the confidence interval.

Upper Confidence Bound (UCB) Algorithm

This algorithm constructs confidence intervals for the estimates of each arm, and chooses the arm with the highest **upper** confidence bound (if the confidence interval is [a,b], we compare only the value of b)

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

---

for $i = 1, 2, \ldots, K$ **do**
- Pull arm i once, observing $r_i \sim Ber(p_i)$.
- Estimate $\hat{p}_i = r_i/1$. ▷ Each estimate $\hat{p}_i$ will initially either be 1 or 0.

for $t = K+1, K+2, \ldots, T$ **do**:
- Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}} \right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
- Receive reward $r_t \sim Ber(p_{a_t})$.

Upper Confidence Bound (UCB) Algorithm

This algorithm constructs confidence intervals for the estimates of each arm, and chooses the arm with the highest **upper** confidence bound (if the confidence interval is [a,b], we compare only the value of b)

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
  Pull arm i once, observing $r_i \sim Ber(p_i)$.
  Estimate $\hat{p}_i = r_i/1$. ▷ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \ldots, T$ **do**:
  Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}} \right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
  Receive reward $r_t \sim Ber(p_{a_t})$.
  Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Upper Confidence Bound (UCB) Algorithm

This algorithm constructs confidence intervals for the estimates of each arm, and chooses the arm with the highest **upper** confidence bound (if the confidence interval is [a,b], we compare only the values of b).

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

---

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

  Pull arm i once, observing $r_i \sim Ber(p_i)$.

  Estimate $\hat{p}_i = r_i/1$. ▹ Each estimate $\hat{p}_i$ will initially either be 1 or 0.

for $t = K+1, K+2, \ldots, T$ **do**:

  Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}} \right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.

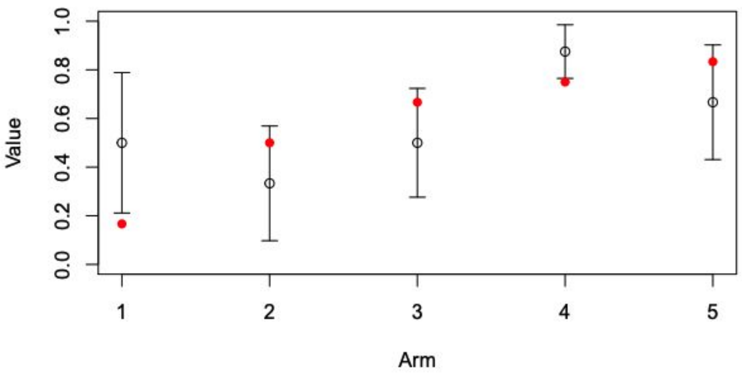
  Receive reward $r_t \sim Ber(p_{a_t})$.

  Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

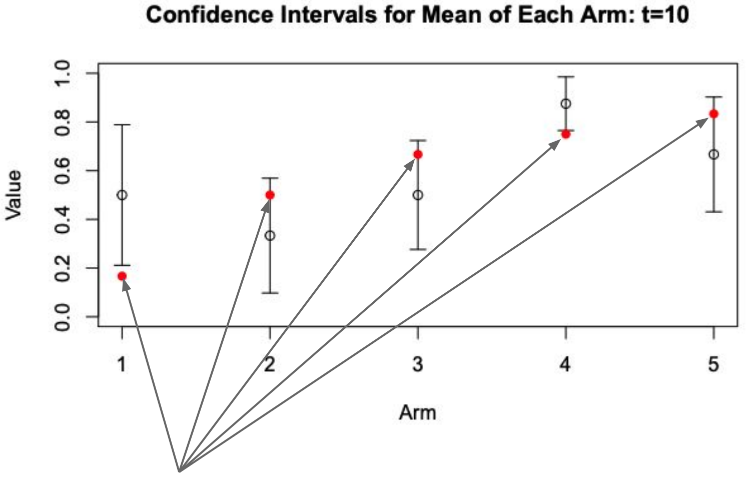
Exploration is "baked in": the frequently pulled arms will have narrow confidence intervals (and hence a lower **upper** bound), and the less-frequently pulled arms will have wide intervals (and hence a **higher** upper bound).

UCB: Confidence Intervals over Time

Confidence Intervals for Mean of Each Arm: t=10

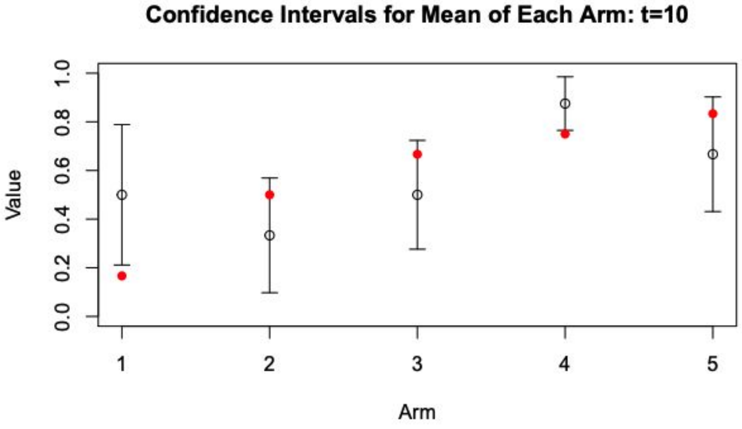


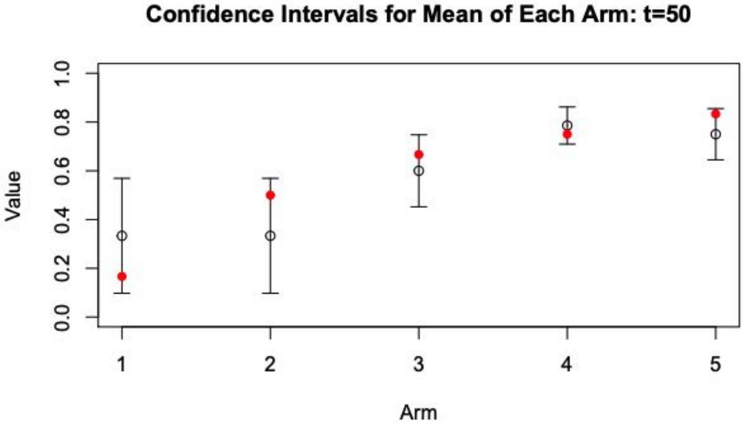
UCB: Confidence Intervals over Time



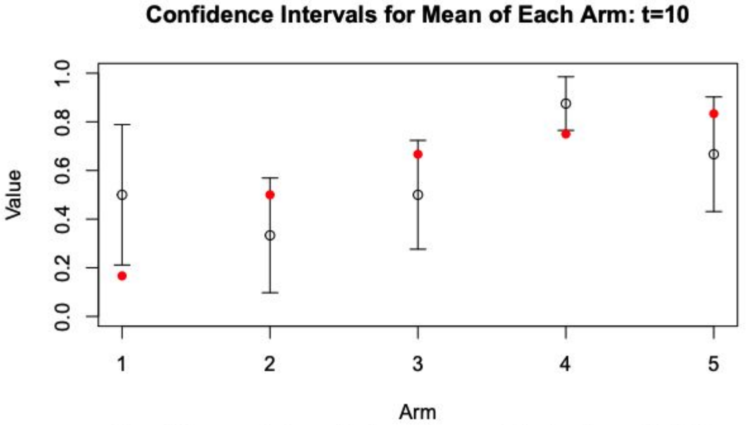
Red Dots: True Means

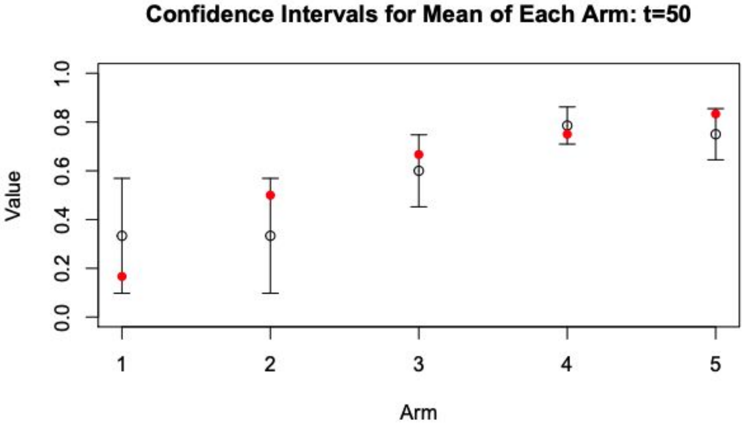
UCB: Confidence Intervals over Time

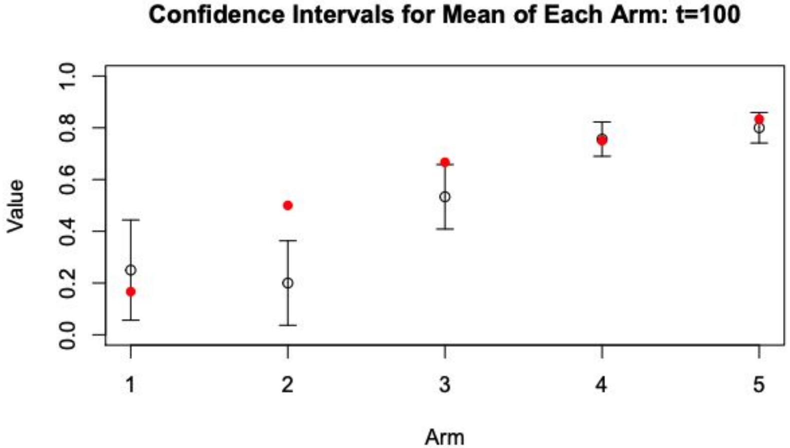




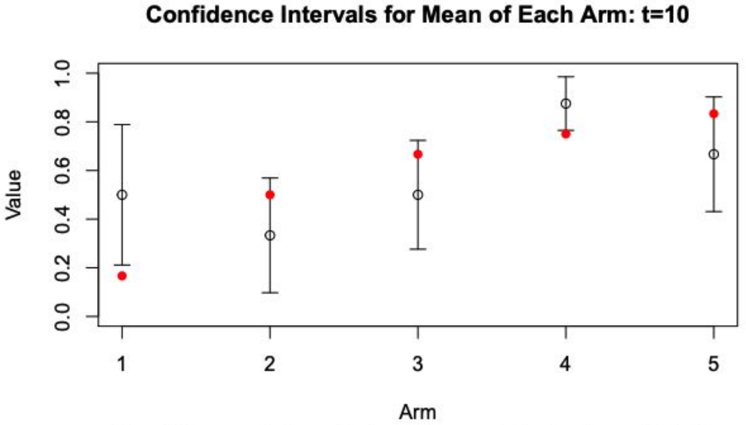
UCB: Confidence Intervals over Time

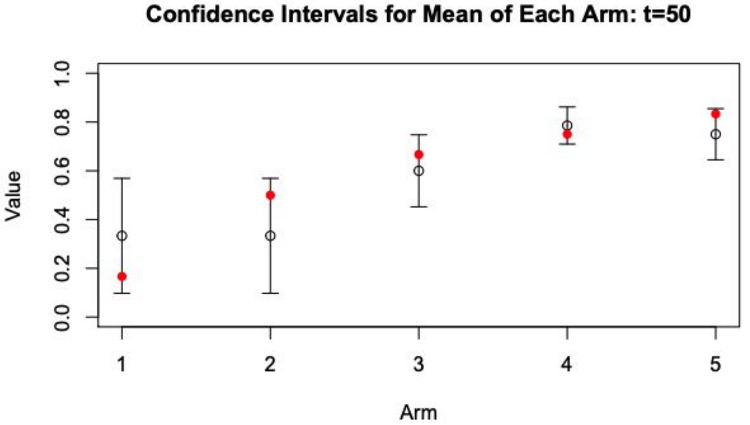


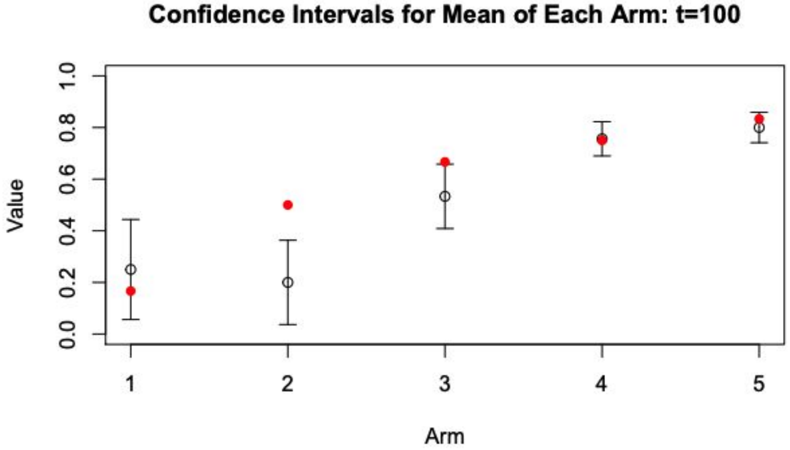


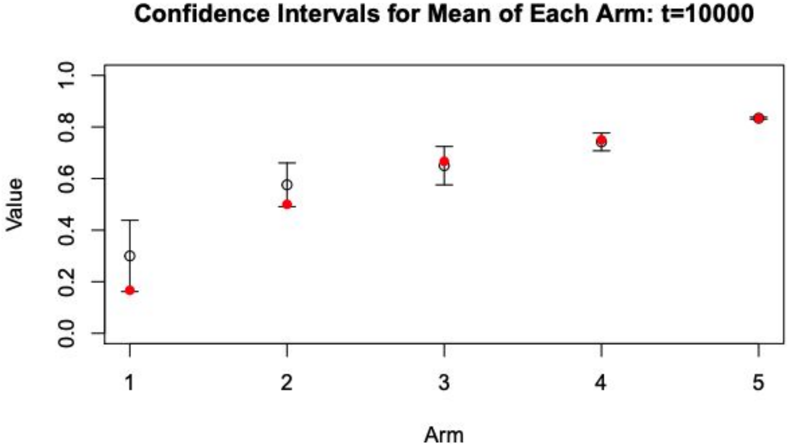


UCB: Confidence Intervals over Time









UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \dots, K$ **do**
    Pull arm i once, observing $r_i \sim Ber(p_i)$.
    Estimate $\hat{p}_i = r_i$. ▹ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \dots, T$ **do:**
    Pull arm $a_t = \arg\max_{i \in \{1,2,\dots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}} \right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
    Receive reward $r_t \sim Ber(p_{a_t})$.
    Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	**UCB** ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5				
2	0.2				
3	0.9				

Time (t)	Arm Pulled (a_t)	Reward (r_t)

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
  Pull arm i once, observing $r_i \sim Ber(p_i)$.
  Estimate $\hat{p}_i = r_i$. ▷ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \ldots, T$ **do**:
  Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}\right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
  Receive reward $r_t \sim Ber(p_{a_t})$.
  Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	**UCB** ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5				
2	0.2				
3	0.9				

Time (t)	Arm Pulled (a_t)	Reward (r_t)

We don't actually know these...

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
    Pull arm i once, observing $r_i \sim Ber(p_i)$.
    Estimate $\hat{p}_i = r_i$.   ▹ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \ldots, T$ **do:**
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}\right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
    Receive reward $r_t \sim Ber(p_{a_t})$.
    Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	UCB ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5				
2	0.2				
3	0.9				

Time (t)	Arm Pulled (a_t)	Reward (r_t)
1	1	0

At time 1, we pull arm 1, and observe either a 1 (with probability 0.5) or a 0 (with probability 1-0.5). We happen to observe a 0.

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
    Pull arm i once, observing $r_i \sim Ber(p_i)$.
    Estimate $\hat{p}_i = r_i$.   ▹ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \ldots, T$ **do**:
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}} \right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
    Receive reward $r_t \sim Ber(p_{a_t})$.
    Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	UCB ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5	1	0	0/1	
2	0.2				
3	0.9				

Time (t)	Arm Pulled (a_t)	Reward (r_t)
1	1	0

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
    Pull arm i once, observing $r_i \sim Ber(p_i)$.
    Estimate $\hat{p}_i = r_i$. ▷ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \ldots, T$ **do:**
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}} \right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
    Receive reward $r_t \sim Ber(p_{a_t})$.
    Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	**UCB** ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5	1	0	0/1	
2	0.2				
3	0.9				

Time (t)	Arm Pulled (a_t)	Reward (r_t)
2	2	0

At time 2, we pull arm 2, and observe either a 1 (with probability 0.2) or a 0 (with probability 1-0.2).
We happen to observe a 0.

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
    Pull arm i once, observing $r_i \sim Ber(p_i)$.
    Estimate $\hat{p}_i = r_i$.   ▷ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \ldots, T$ **do**:
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}} \right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
    Receive reward $r_t \sim Ber(p_{a_t})$.
    Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	UCB ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5	1	0	0/1	
2	0.2	1	0	0/1	
3	0.9				

Time (t)	Arm Pulled (a_t)	Reward (r_t)
2	2	0

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
- Pull arm i once, observing $r_i \sim Ber(p_i)$.
- Estimate $\hat{p}_i = r_i$. ▷ Each estimate $\hat{p}_i$ will initially either be 1 or 0.

for $t = K+1, K+2, \ldots, T$ **do**:
- Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}} \right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
- Receive reward $r_t \sim Ber(p_{a_t})$.
- Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	**UCB** ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5	1	0	0/1	
2	0.2	1	0	0/1	
3	0.9				

Time (t)	Arm Pulled (a_t)	Reward (r_t)
3	3	1

At time 3, we pull arm 3, and observe either a 1 (with probability 0.9) or a 0 (with probability 1-0.9). We happen to observe a 1.

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
    Pull arm i once, observing $r_i \sim Ber(p_i)$.
    Estimate $\hat{p}_i = r_i$.   ▷ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \ldots, T$ **do**:
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}} \right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
    Receive reward $r_t \sim Ber(p_{a_t})$.
    Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	**UCB** ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5	1	0	0/1	
2	0.2	1	0	0/1	
3	0.9	1	1	1/1	

Time (t)	Arm Pulled (a_t)	Reward (r_t)
3	3	1

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
    Pull arm i once, observing $r_i \sim Ber(p_i)$.
    Estimate $\hat{p}_i = r_i$. ▷ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \ldots, T$ **do:**
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}} \right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
    Receive reward $r_t \sim Ber(p_{a_t})$.
    Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	UCB ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5	1	0	0/1	
2	0.2	1	0	0/1	
3	0.9	1	1	1/1	

Time (t)	Arm Pulled (a_t)	Reward (r_t)
4		

At time 4, we must compute all our upper confidence bounds, and choose the best one.

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
  Pull arm i once, observing $r_i \sim Ber(p_i)$.
  Estimate $\hat{p}_i = r_i$. ▹ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \ldots, T$ **do**:
  Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}} \right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
  Receive reward $r_t \sim Ber(p_{a_t})$.
  Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	UCB ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5	1	0	0/1	1.665
2	0.2	1	0	0/1	
3	0.9	1	1	1/1	

Time (t)	Arm Pulled (a_t)	Reward (r_t)
4		

$$0 + \sqrt{\frac{2\ln(4)}{1}} \approx 1.665$$

At time 4, we must compute all our upper confidence bounds, and choose the best one.

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
  Pull arm i once, observing $r_i \sim Ber(p_i)$.
  Estimate $\hat{p}_i = r_i$. ▷ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \ldots, T$ **do**:
  Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}\right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
  Receive reward $r_t \sim Ber(p_{a_t})$.
  Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	UCB ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5	1	0	0/1	1.665
2	0.2	1	0	0/1	1.665
3	0.9	1	1	1/1	

Time (t)	Arm Pulled (a_t)	Reward (r_t)
4		

$$0 + \sqrt{\frac{2\ln(4)}{1}} \approx 1.665$$

At time 4, we must compute all our upper confidence bounds, and choose the best one.

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

```
for i = 1, 2, ..., K do
    Pull arm i once, observing r_i ~ Ber(p_i).
    Estimate p̂_i = r_i.                      ▹ Each estimate p̂_i will initially either be 1 or 0.
for t = K + 1, K + 2, ..., T do:
    Pull arm a_t = arg max_{i∈{1,2,...,K}} ( p̂_i + sqrt( 2 ln(t) / N_t(i) ) ), where N_t(i) is the number of times arm i was pulled before time t.
    Receive reward r_t ~ Ber(p_{a_t}).
    Update N_t(a_t) and p̂_{a_t} (using newly observed reward r_t).
```

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	**UCB** ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5	1	0	0/1	1.665
2	0.2	1	0	0/1	1.665
3	0.9	1	1	1/1	2.665

Time (t)	Arm Pulled (a_t)	Reward (r_t)
4		

$$1 + \sqrt{\frac{2\ln(4)}{1}} \approx 2.665$$

At time 4, we must compute all our upper confidence bounds, and choose the best one.

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
  Pull arm i once, observing $r_i \sim Ber(p_i)$.
  Estimate $\hat{p}_i = r_i$. ▹ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \ldots, T$ **do**:
  Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}\right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
  Receive reward $r_t \sim Ber(p_{a_t})$.
  Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	UCB ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5	1	0	0/1	1.665
2	0.2	1	0	0/1	1.665
3	0.9	1	1	1/1	2.665

Time (t)	Arm Pulled (a_t)	Reward (r_t)
4	3	

At time 4, arm 3 has the highest UCB so we pull it.

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

1: **for** $i = 1, 2, \ldots, K$ **do**
2: Pull arm i once, observing $r_i \sim Ber(p_i)$.
3: Estimate $\hat{p}_i = r_i$. ▷ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
4: **for** $t = K+1, K+2, \ldots, T$ **do**:
5: Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}} \right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
6: Receive reward $r_t \sim Ber(p_{a_t})$.
7: Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	**UCB** ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5	1	0	0/1	1.665
2	0.2	1	0	0/1	1.665
3	0.9	1	1	1/1	2.665

Time (t)	Arm Pulled (a_t)	Reward (r_t)
4	3	0

At time 4, arm 3 has the highest UCB so we pull it. We observe a reward of 0.

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

1: **for** $i = 1, 2, \ldots, K$ **do**
2: Pull arm i once, observing $r_i \sim Ber(p_i)$.
3: Estimate $\hat{p}_i = r_i$. ▷ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
4: **for** $t = K+1, K+2, \ldots, T$ **do**:
5: Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}\right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
6: Receive reward $r_t \sim Ber(p_{a_t})$.
7: Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	UCB ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5	1	0	0/1	
2	0.2	1	0	0/1	
3	0.9	2	1	1/2	

Time (t)	Arm Pulled (a_t)	Reward (r_t)
4	3	0

At time 4, arm 3 has the highest UCB so we pull it. We observe a reward of 0. Then we update our estimate for p_3.

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
  Pull arm i once, observing $r_i \sim Ber(p_i)$.
  Estimate $\hat{p}_i = r_i$. ▷ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K + 1, K + 2, \ldots, T$ **do:**
  Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}} \right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
  Receive reward $r_t \sim Ber(p_{a_t})$.
  Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	**UCB** ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5	1	0	0/1	
2	0.2	1	0	0/1	
3	0.9	2	1	1/2	

Time (t)	Arm Pulled (a_t)	Reward (r_t)
5		

At time 5, we must compute all our upper confidence bounds, and choose the best one.

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
    Pull arm i once, observing $r_i \sim Ber(p_i)$.
    Estimate $\hat{p}_i = r_i$. ▷ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \ldots, T$ **do**:
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}\right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
    Receive reward $r_t \sim Ber(p_{a_t})$.
    Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	UCB ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5	1	0	0/1	1.794
2	0.2	1	0	0/1	
3	0.9	2	1	1/2	

Time (t)	Arm Pulled (a_t)	Reward (r_t)
5		

$$0 + \sqrt{\frac{2\ln(5)}{1}} \approx 1.794$$

At time 5, we must compute all our upper confidence bounds, and choose the best one.

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
  Pull arm i once, observing $r_i \sim Ber(p_i)$.
  Estimate $\hat{p}_i = r_i$. ▷ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \ldots, T$ **do**:
  Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}} \right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
  Receive reward $r_t \sim Ber(p_{a_t})$.
  Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	**UCB** ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5	1	0	0/1	1.794
2	0.2	1	0	0/1	1.794
3	0.9	2	1	1/2	

Time (t)	Arm Pulled (a_t)	Reward (r_t)
5		

$$0 + \sqrt{\frac{2\ln(5)}{1}} \approx 1.794$$

At time 5, we must compute all our upper confidence bounds, and choose the best one.

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
  Pull arm i once, observing $r_i \sim Ber(p_i)$.
  Estimate $\hat{p}_i = r_i$. ▹ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \ldots, T$ **do**:
  Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}} \right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
  Receive reward $r_t \sim Ber(p_{a_t})$.
  Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	**UCB** ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5	1	0	0/1	1.794
2	0.2	1	0	0/1	1.794
3	0.9	2	1	1/2	1.769

Time (t)	Arm Pulled (a_t)	Reward (r_t)
5		

$$\frac{1}{2} + \sqrt{\frac{2\ln(5)}{2}} \approx 1.769$$

At time 5, we must compute all our upper confidence bounds, and choose the best one.

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
  Pull arm i once, observing $r_i \sim Ber(p_i)$.
  Estimate $\hat{p}_i = r_i$. ▹ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \ldots, T$ **do**:
  Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}\right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
  Receive reward $r_t \sim Ber(p_{a_t})$.
  Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	UCB ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5	1	0	0/1	1.794
2	0.2	1	0	0/1	1.794
3	0.9	2	1	1/2	1.769

Time (t)	Arm Pulled (a_t)	Reward (r_t)
5	1	

At time 5, arms 1 and 2 have the highest UCB so we pull one of them (let's break ties by choosing the smaller index arm). So we pull arm 1.

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
  Pull arm i once, observing $r_i \sim Ber(p_i)$.
  Estimate $\hat{p}_i = r_i$. ▹ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \ldots, T$ **do:**
  Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}\right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
  Receive reward $r_t \sim Ber(p_{a_t})$.
  Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	UCB ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5	1	0	0/1	1.794
2	0.2	1	0	0/1	1.794
3	0.9	2	1	1/2	1.769

Time (t)	Arm Pulled (a_t)	Reward (r_t)
5	1	0

At time 5, arms 1 and 2 have the highest UCB so we pull one of them (let's break ties by choosing the smaller index arm). So we pull arm 1.
We observe a reward of 0.

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

for $i = 1, 2, \ldots, K$ **do**
  Pull arm i once, observing $r_i \sim Ber(p_i)$.
  Estimate $\hat{p}_i = r_i$. ▹ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \ldots, T$ **do**:
  Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}} \right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
  Receive reward $r_t \sim Ber(p_{a_t})$.
  Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	**UCB** ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5	2	0	0/2	
2	0.2	1	0	0/1	
3	0.9	2	1	1/2	

Time (t)	Arm Pulled (a_t)	Reward (r_t)
5	1	0

At time 5, arms 1 and 2 have the highest UCB so we pull one of them (let's break ties by choosing the smaller index arm). So we pull arm 1.
We observe a reward of 0. Then we update our estimate for p_1.

UCB Example

Algorithm 4 UCB1 Algorithm (Upper Confidence Bound) for Bernoulli Bandits

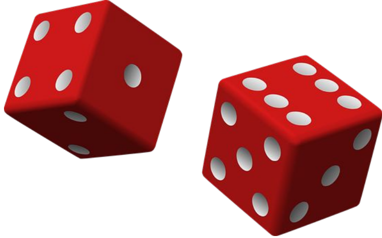
for $i = 1, 2, \ldots, K$ **do**
    Pull arm i once, observing $r_i \sim Ber(p_i)$.
    Estimate $\hat{p}_i = r_i$. ▹ Each estimate $\hat{p}_i$ will initially either be 1 or 0.
for $t = K+1, K+2, \ldots, T$ **do**:
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} \left(\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}} \right)$, where $N_t(i)$ is the number of times arm i was pulled before time t.
    Receive reward $r_t \sim Ber(p_{a_t})$.
    Update $N_t(a_t)$ and $\hat{p}_{a_t}$ (using newly observed reward r_t).

Arm (i)	True p_i	# Times Pulled	Total Reward	$\hat{p}_i$	**UCB** ($\hat{p}_i + \sqrt{\frac{2\ln(t)}{N_t(i)}}$)
1	0.5	2	0	0/2	
2	0.2	1	0	0/1	
3	0.9	2	1	1/2	

Time (t)	Arm Pulled (a_t)	Reward (r_t)
6		

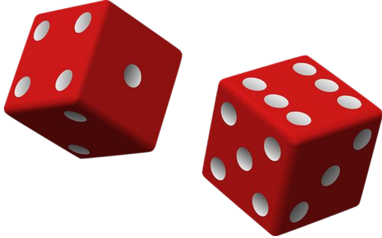
And so on!!! Notice how we started exploring since the confidence bound grows with t for even the unexplored arms!

Thompson Sampling Algorithm



Use MAP: Assume a Beta(1,1) (Uniform) prior on each unknown probability of reward.

Thompson Sampling Algorithm

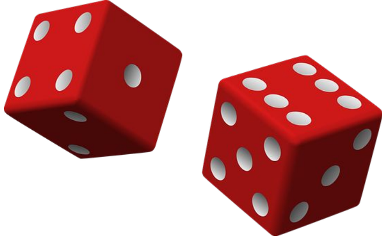


Use MAP: Assume a Beta(1,1) (Uniform) prior on each unknown probability of reward.

Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

1: For each arm $i \in \{1, \ldots, K\}$, initialize $\alpha_i = \beta_i = 1$. ▷ Set $Beta(\alpha_i, \beta_i)$ prior for each p_i.

Thompson Sampling Algorithm

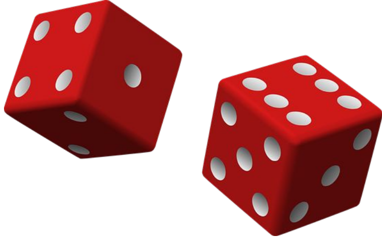


Use MAP: Assume a Beta(1,1) (Uniform) prior on each unknown probability of reward.

Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

For each arm $i \in \{1, \ldots, K\}$, initialize $\alpha_i = \beta_i = 1$. ▹ Set $Beta(\alpha_i, \beta_i)$ prior for each p_i.
for $t = 1, 2, \ldots, T$ **do**:
    For each arm i, get sample $s_{i,t} \sim Beta(\alpha_i, \beta_i)$. ▹ Each is a float in $[0, 1]$.

Thompson Sampling Algorithm



Use MAP: Assume a Beta(1,1) (Uniform) prior on each unknown probability of reward.

Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

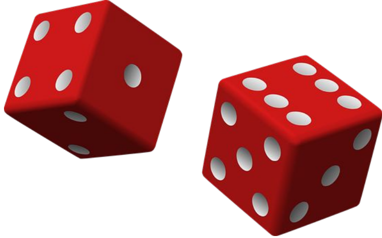
For each arm $i \in \{1, \ldots, K\}$, initialize $\alpha_i = \beta_i = 1$. ▷ Set $Beta(\alpha_i, \beta_i)$ prior for each p_i.

for $t = 1, 2, \ldots, T$ **do**:

    For each arm i, get sample $s_{i,t} \sim Beta(\alpha_i, \beta_i)$. ▷ Each is a float in $[0, 1]$.

    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} s_{i,t}$. ▷ This "bakes in" exploration!

Thompson Sampling Algorithm

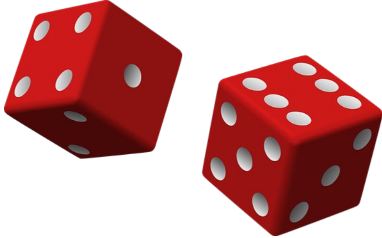


Use MAP: Assume a Beta(1,1) (Uniform) prior on each unknown probability of reward.

Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

For each arm $i \in \{1, \ldots, K\}$, initialize $\alpha_i = \beta_i = 1$. ▹ Set $Beta(\alpha_i, \beta_i)$ prior for each p_i.
for $t = 1, 2, \ldots, T$ **do**:
    For each arm i, get sample $s_{i,t} \sim Beta(\alpha_i, \beta_i)$. ▹ Each is a float in $[0, 1]$.
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} s_{i,t}$. ▹ This "bakes in" exploration!
    Receive reward $r_t \sim Ber(p_{a_t})$.

Thompson Sampling Algorithm



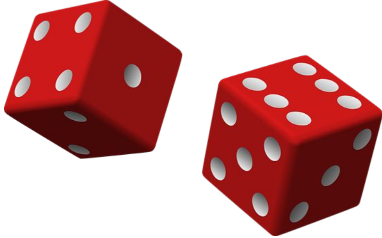
Use MAP: Assume a Beta(1,1) (Uniform) prior on each unknown probability of reward.

Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

---

For each arm $i \in \{1, \ldots, K\}$, initialize $\alpha_i = \beta_i = 1$. ▷ Set $Beta(\alpha_i, \beta_i)$ prior for each p_i.
for $t = 1, 2, \ldots, T$ **do**:
    For each arm i, get sample $s_{i,t} \sim Beta(\alpha_i, \beta_i)$. ▷ Each is a float in $[0, 1]$.
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} s_{i,t}$. ▷ This "bakes in" exploration!
    Receive reward $r_t \sim Ber(p_{a_t})$.
    if $r_t == 1$ **then** $\alpha_{a_t} \leftarrow \alpha_{a_t} + 1$. ▷ Increment number of "successes".
    else if $r_t == 0$ **then** $\beta_{a_t} \leftarrow \beta_{a_t} + 1$. ▷ Increment number of "failures".

Thompson Sampling Algorithm



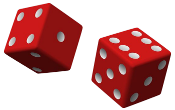
Use MAP: Assume a Beta(1,1) (Uniform) prior on each unknown probability of reward.

Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

For each arm $i \in \{1, \ldots, K\}$, initialize $\alpha_i = \beta_i = 1$. ▹ Set $Beta(\alpha_i, \beta_i)$ prior for each p_i.
for $t = 1, 2, \ldots, T$ **do**:
    For each arm i, get sample $s_{i,t} \sim Beta(\alpha_i, \beta_i)$. ▹ Each is a float in $[0, 1]$.
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} s_{i,t}$. ▹ This "bakes in" exploration!
    Receive reward $r_t \sim Ber(p_{a_t})$.
    if $r_t == 1$ **then** $\alpha_{a_t} \leftarrow \alpha_{a_t} + 1$. ▹ Increment number of "successes".
    else if $r_t == 0$ **then** $\beta_{a_t} \leftarrow \beta_{a_t} + 1$. ▹ Increment number of "failures".

The exploration comes in since we **sample** from each Beta distribution, rather than just choosing the one with largest expectation or mode (greedy).

Thompson Example



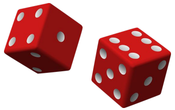
Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

For each arm $i \in \{1, \ldots, K\}$, initialize $\alpha_i = \beta_i = 1$. ▷ Set $Beta(\alpha_i, \beta_i)$ prior for each p_i.
for $t = 1, 2, \ldots, T$ **do**:
    For each arm i, get sample $s_{i,t} \sim Beta(\alpha_i, \beta_i)$. ▷ Each is a float in $[0, 1]$.
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} s_{i,t}$. ▷ This "bakes in" exploration!
    Receive reward $r_t \sim Ber(p_{a_t})$.
    if $r_t == 1$ **then** $\alpha_{a_t} \leftarrow \alpha_{a_t} + 1$. ▷ Increment number of "successes".
    else if $r_t == 0$ **then** $\beta_{a_t} \leftarrow \beta_{a_t} + 1$. ▷ Increment number of "failures".

Arm (i)	True p_i	α_i	β_i	$s_{i,t}$
1	0.5			
2	0.2			
3	0.9			

Time (t)	Arm Pulled (a_t)	Reward (r_t)

Thompson Example



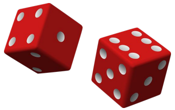
Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

For each arm $i \in \{1, \ldots, K\}$, initialize $\alpha_i = \beta_i = 1$. ▹ Set $Beta(\alpha_i, \beta_i)$ prior for each p_i.
for $t = 1, 2, \ldots, T$ **do**:
    For each arm i, get sample $s_{i,t} \sim Beta(\alpha_i, \beta_i)$. ▹ Each is a float in $[0, 1]$.
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} s_{i,t}$. ▹ This "bakes in" exploration!
    Receive reward $r_t \sim Ber(p_{a_t})$.
    if $r_t == 1$ **then** $\alpha_{a_t} \leftarrow \alpha_{a_t} + 1$. ▹ Increment number of "successes".
    else if $r_t == 0$ **then** $\beta_{a_t} \leftarrow \beta_{a_t} + 1$. ▹ Increment number of "failures".

Arm (i)	True p_i	α_i	β_i	$s_{i,t}$
1	0.5	1	1	
2	0.2	1	1	
3	0.9	1	1	

Time (t)	Arm Pulled (a_t)	Reward (r_t)
1		

Thompson Example



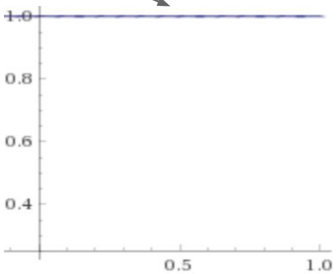
Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

```
For each arm i ∈ {1, ..., K}, initialize α_i = β_i = 1.          ▷ Set Beta(α_i, β_i) prior for each p_i.
for t = 1, 2, ..., T do:
    For each arm i, get sample s_{i,t} ~ Beta(α_i, β_i).        ▷ Each is a float in [0, 1].
    Pull arm a_t = arg max_{i∈{1,2,...,K}} s_{i,t}.               ▷ This "bakes in" exploration!
    Receive reward r_t ~ Ber(p_{a_t}).
    if r_t == 1 then α_{a_t} ← α_{a_t} + 1.                      ▷ Increment number of "successes".
    else if r_t == 0 then β_{a_t} ← β_{a_t} + 1.                 ▷ Increment number of "failures".
```

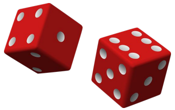
Arm (i)	True p_i	α_i	β_i	$s_{i,t}$
1	0.5	1	1	0.43
2	0.2	1	1	
3	0.9	1	1	

Time (t)	Arm Pulled (a_t)	Reward (r_t)
1		

Sample from Beta(1,1) density →



Thompson Example



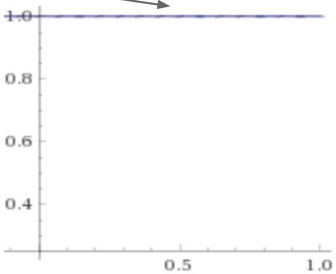
Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

For each arm $i \in \{1, \ldots, K\}$, initialize $\alpha_i = \beta_i = 1$. ▷ Set $Beta(\alpha_i, \beta_i)$ prior for each p_i.
for $t = 1, 2, \ldots, T$ **do**:
    For each arm i, get sample $s_{i,t} \sim Beta(\alpha_i, \beta_i)$. ▷ Each is a float in $[0, 1]$.
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} s_{i,t}$. ▷ This "bakes in" exploration!
    Receive reward $r_t \sim Ber(p_{a_t})$.
    if $r_t == 1$ **then** $\alpha_{a_t} \leftarrow \alpha_{a_t} + 1$. ▷ Increment number of "successes".
    else if $r_t == 0$ **then** $\beta_{a_t} \leftarrow \beta_{a_t} + 1$. ▷ Increment number of "failures".

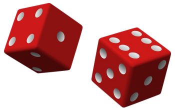
Arm (i)	True p_i	α_i	β_i	$s_{i,t}$
1	0.5	1	1	0.43
2	0.2	1	1	0.75
3	0.9	1	1	

Time (t)	Arm Pulled (a_t)	Reward (r_t)
1		

Sample from Beta(1,1) density →



Thompson Example

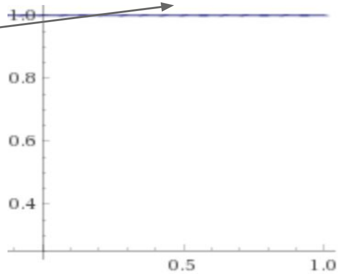


Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

For each arm $i \in \{1, \ldots, K\}$, initialize $\alpha_i = \beta_i = 1$. ▷ Set $Beta(\alpha_i, \beta_i)$ prior for each p_i.
for $t = 1, 2, \ldots, T$ **do**:
  For each arm i, get sample $s_{i,t} \sim Beta(\alpha_i, \beta_i)$. ▷ Each is a float in $[0, 1]$.
  Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} s_{i,t}$. ▷ This "bakes in" exploration!
  Receive reward $r_t \sim Ber(p_{a_t})$.
  if $r_t == 1$ **then** $\alpha_{a_t} \leftarrow \alpha_{a_t} + 1$. ▷ Increment number of "successes".
  else if $r_t == 0$ **then** $\beta_{a_t} \leftarrow \beta_{a_t} + 1$. ▷ Increment number of "failures".

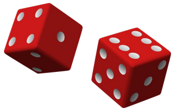
Arm (i)	True p_i	α_i	β_i	$s_{i,t}$
1	0.5	1	1	0.43
2	0.2	1	1	0.75
3	0.9	1	1	0.11

Sample from Beta(1,1) density →



Time (t)	Arm Pulled (a_t)	Reward (r_t)
1		

Thompson Example



Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

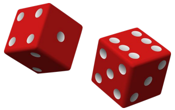
For each arm $i \in \{1, \ldots, K\}$, initialize $\alpha_i = \beta_i = 1$. ▹ Set $Beta(\alpha_i, \beta_i)$ prior for each p_i.
for $t = 1, 2, \ldots, T$ **do**:
    For each arm i, get sample $s_{i,t} \sim Beta(\alpha_i, \beta_i)$. ▹ Each is a float in $[0, 1]$.
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} s_{i,t}$. ▹ This "bakes in" exploration!
    Receive reward $r_t \sim Ber(p_{a_t})$.
    if $r_t == 1$ **then** $\alpha_{a_t} \leftarrow \alpha_{a_t} + 1$. ▹ Increment number of "successes".
    else if $r_t == 0$ **then** $\beta_{a_t} \leftarrow \beta_{a_t} + 1$. ▹ Increment number of "failures".

Arm (i)	True p_i	α_i	β_i	$s_{i,t}$
1	0.5	1	1	0.43
2	0.2	1	1	0.75
3	0.9	1	1	0.11

Time (t)	Arm Pulled (a_t)	Reward (r_t)
1	2	

Choose arm with highest sample!

Thompson Example



Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

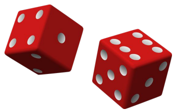
```
For each arm i ∈ {1, ..., K}, initialize α_i = β_i = 1.        ▹ Set Beta(α_i, β_i) prior for each p_i.
for t = 1, 2, ..., T do:
    For each arm i, get sample s_{i,t} ~ Beta(α_i, β_i).        ▹ Each is a float in [0, 1].
    Pull arm a_t = arg max_{i∈{1,2,...,K}} s_{i,t}.              ▹ This "bakes in" exploration!
    Receive reward r_t ~ Ber(p_{a_t}).
    if r_t == 1 then α_{a_t} ← α_{a_t} + 1.                     ▹ Increment number of "successes".
    else if r_t == 0 then β_{a_t} ← β_{a_t} + 1.                ▹ Increment number of "failures".
```

Arm (i)	True p_i	α_i	β_i	$s_{i,t}$
1	0.5	1	1	0.43
2	0.2	1	1	0.75
3	0.9	1	1	0.11

Time (t)	Arm Pulled (a_t)	Reward (r_t)
1	2	0

Observe reward 1 with probability 0.2 and 0 with probability 0.8.

Thompson Example



Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

For each arm $i \in \{1, \dots, K\}$, initialize $\alpha_i = \beta_i = 1$. ▹ Set $Beta(\alpha_i, \beta_i)$ prior for each p_i.
for $t = 1, 2, \dots, T$ **do**:
  For each arm i, get sample $s_{i,t} \sim Beta(\alpha_i, \beta_i)$. ▹ Each is a float in $[0, 1]$.
  Pull arm $a_t = \arg\max_{i \in \{1,2,\dots,K\}} s_{i,t}$. ▹ This "bakes in" exploration!
  Receive reward $r_t \sim Ber(p_{a_t})$.
  if $r_t == 1$ **then** $\alpha_{a_t} \leftarrow \alpha_{a_t} + 1$. ▹ Increment number of "successes".
  else if $r_t == 0$ **then** $\beta_{a_t} \leftarrow \beta_{a_t} + 1$. ▹ Increment number of "failures".

Arm (i)	True p_i	α_i	β_i	$s_{i,t}$
1	0.5	1	1	
2	0.2	1	2	
3	0.9	1	1	

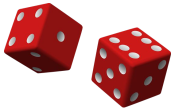
Time (t)	Arm Pulled (a_t)	Reward (r_t)
1	2	0

Add a count of 1 to the failures :(.

Thompson Example

Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

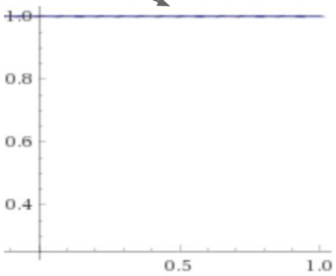
For each arm $i \in \{1, \ldots, K\}$, initialize $\alpha_i = \beta_i = 1$. ▷ Set $Beta(\alpha_i, \beta_i)$ prior for each p_i.
for $t = 1, 2, \ldots, T$ **do**:
    For each arm i, get sample $s_{i,t} \sim Beta(\alpha_i, \beta_i)$. ▷ Each is a float in $[0, 1]$.
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} s_{i,t}$. ▷ This "bakes in" exploration!
    Receive reward $r_t \sim Ber(p_{a_t})$.
    if $r_t == 1$ **then** $\alpha_{a_t} \leftarrow \alpha_{a_t} + 1$. ▷ Increment number of "successes".
    else if $r_t == 0$ **then** $\beta_{a_t} \leftarrow \beta_{a_t} + 1$. ▷ Increment number of "failures".



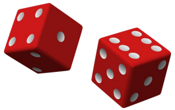
Arm (i)	True p_i	α_i	β_i	$s_{i,t}$
1	0.5	1	1	0.52
2	0.2	1	2	
3	0.9	1	1	

Time (t)	Arm Pulled (a_t)	Reward (r_t)
2		

Sample from Beta(1,1) density →



Thompson Example



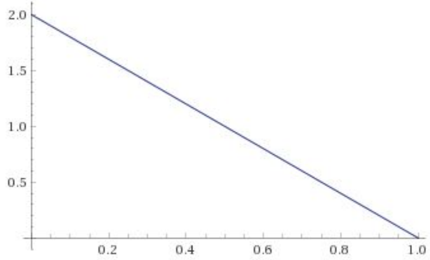
Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

For each arm $i \in \{1, \dots, K\}$, initialize $\alpha_i = \beta_i = 1$. ▹ Set $Beta(\alpha_i, \beta_i)$ prior for each p_i.
for $t = 1, 2, \dots, T$ **do**:
  For each arm i, get sample $s_{i,t} \sim Beta(\alpha_i, \beta_i)$. ▹ Each is a float in $[0, 1]$.
  Pull arm $a_t = \arg\max_{i \in \{1,2,\dots,K\}} s_{i,t}$. ▹ This "bakes in" exploration!
  Receive reward $r_t \sim Ber(p_{a_t})$.
  if $r_t == 1$ **then** $\alpha_{a_t} \leftarrow \alpha_{a_t} + 1$. ▹ Increment number of "successes".
  else if $r_t == 0$ **then** $\beta_{a_t} \leftarrow \beta_{a_t} + 1$. ▹ Increment number of "failures".

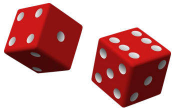
Arm (i)	True p_i	α_i	β_i	$s_{i,t}$
1	0.5	1	1	0.52
2	0.2	1	2	0.05
3	0.9	1	1	

Time (t)	Arm Pulled (a_t)	Reward (r_t)
2		

Sample from Beta(1,2) density →



Thompson Example



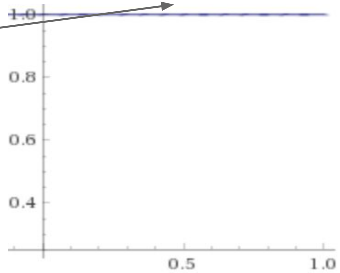
Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

For each arm $i \in \{1, \ldots, K\}$, initialize $\alpha_i = \beta_i = 1$. ▷ Set $Beta(\alpha_i, \beta_i)$ prior for each p_i.
for $t = 1, 2, \ldots, T$ **do**:
  For each arm i, get sample $s_{i,t} \sim Beta(\alpha_i, \beta_i)$. ▷ Each is a float in $[0, 1]$.
  Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} s_{i,t}$. ▷ This "bakes in" exploration!
  Receive reward $r_t \sim Ber(p_{a_t})$.
  if $r_t == 1$ **then** $\alpha_{a_t} \leftarrow \alpha_{a_t} + 1$. ▷ Increment number of "successes".
  else if $r_t == 0$ **then** $\beta_{a_t} \leftarrow \beta_{a_t} + 1$. ▷ Increment number of "failures".

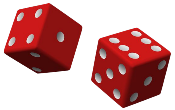
Arm (i)	True p_i	α_i	β_i	$s_{i,t}$
1	0.5	1	1	0.52
2	0.2	1	2	0.05
3	0.9	1	1	0.67

Time (t)	Arm Pulled (a_t)	Reward (r_t)
2		

Sample from Beta(1,1) density →



Thompson Example



Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

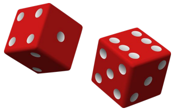
For each arm $i \in \{1, \ldots, K\}$, initialize $\alpha_i = \beta_i = 1$. ▹ Set $Beta(\alpha_i, \beta_i)$ prior for each p_i.
for $t = 1, 2, \ldots, T$ **do**:
    For each arm i, get sample $s_{i,t} \sim Beta(\alpha_i, \beta_i)$. ▹ Each is a float in $[0, 1]$.
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} s_{i,t}$. ▹ This "bakes in" exploration!
    Receive reward $r_t \sim Ber(p_{a_t})$.
    if $r_t == 1$ **then** $\alpha_{a_t} \leftarrow \alpha_{a_t} + 1$. ▹ Increment number of "successes".
    else if $r_t == 0$ **then** $\beta_{a_t} \leftarrow \beta_{a_t} + 1$. ▹ Increment number of "failures".

Arm (i)	True p_i	α_i	β_i	$s_{i,t}$
1	0.5	1	1	0.52
2	0.2	1	2	0.05
3	0.9	1	1	0.67

Time (t)	Arm Pulled (a_t)	Reward (r_t)
2	3	

Choose arm with highest sample!

Thompson Example



Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

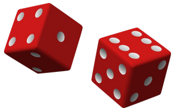
```
For each arm i ∈ {1, ..., K}, initialize α_i = β_i = 1.      ▹ Set Beta(α_i, β_i) prior for each p_i.
for t = 1, 2, ..., T do:
    For each arm i, get sample s_{i,t} ~ Beta(α_i, β_i).      ▹ Each is a float in [0, 1].
    Pull arm a_t = arg max_{i∈{1,2,...,K}} s_{i,t}.           ▹ This "bakes in" exploration!
    Receive reward r_t ~ Ber(p_{a_t}).
    if r_t == 1 then α_{a_t} ← α_{a_t} + 1.                   ▹ Increment number of "successes".
    else if r_t == 0 then β_{a_t} ← β_{a_t} + 1.              ▹ Increment number of "failures".
```

Arm (i)	True p_i	α_i	β_i	$s_{i,t}$
1	0.5	1	1	0.52
2	0.2	1	2	0.05
3	0.9	1	1	0.67

Time (t)	Arm Pulled (a_t)	Reward (r_t)
2	3	1

Observe reward 1 with probability 0.9 and 0 with probability 0.1.

Thompson Example



Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

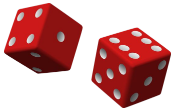
For each arm $i \in \{1, \ldots, K\}$, initialize $\alpha_i = \beta_i = 1$. ▷ Set $Beta(\alpha_i, \beta_i)$ prior for each p_i.
for $t = 1, 2, \ldots, T$ **do**:
    For each arm i, get sample $s_{i,t} \sim Beta(\alpha_i, \beta_i)$. ▷ Each is a float in $[0, 1]$.
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} s_{i,t}$. ▷ This "bakes in" exploration!
    Receive reward $r_t \sim Ber(p_{a_t})$.
    if $r_t == 1$ **then** $\alpha_{a_t} \leftarrow \alpha_{a_t} + 1$. ▷ Increment number of "successes".
    else if $r_t == 0$ **then** $\beta_{a_t} \leftarrow \beta_{a_t} + 1$. ▷ Increment number of "failures".

Arm (i)	True p_i	α_i	β_i	$s_{i,t}$
1	0.5	1	1	
2	0.2	1	2	
3	0.9	2	1	

Time (t)	Arm Pulled (a_t)	Reward (r_t)
2	3	1

Add a count of 1 to the successes :).

Thompson Example



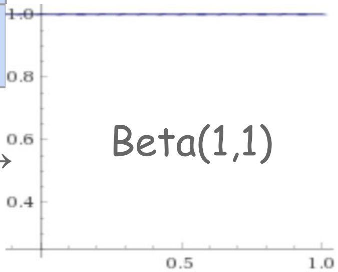
Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

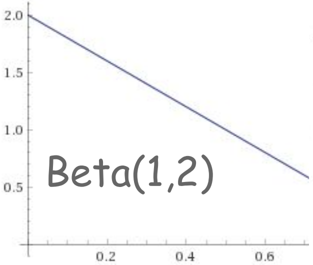
```
For each arm i ∈ {1, . . . , K}, initialize α_i = β_i = 1.        ▷ Set Beta(α_i, β_i) prior for each p_i.
for t = 1, 2, . . . , T do:
    For each arm i, get sample s_{i,t} ~ Beta(α_i, β_i).         ▷ Each is a float in [0, 1].
    Pull arm a_t = arg max_{i∈{1,2,...,K}} s_{i,t}.               ▷ This "bakes in" exploration!
    Receive reward r_t ~ Ber(p_{a_t}).
    if r_t == 1 then α_{a_t} ← α_{a_t} + 1.                      ▷ Increment number of "successes".
    else if r_t == 0 then β_{a_t} ← β_{a_t} + 1.                 ▷ Increment number of "failures".
```

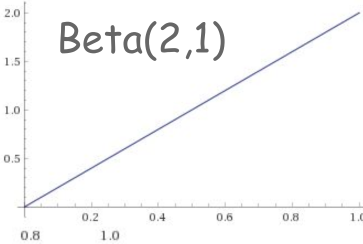
Arm (i)	True p_i	α_i	β_i	$s_{i,t}$
1	0.5	1	1	0.44
2	0.2	1	2	0.27
3	0.9	2	1	0.86

Time (t)	Arm Pulled (a_t)	Reward (r_t)
3		

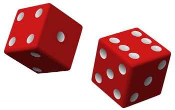
Sample from each arm's Beta distribution →







Thompson Example



Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

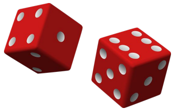
For each arm $i \in \{1, \ldots, K\}$, initialize $\alpha_i = \beta_i = 1$. ▹ Set $Beta(\alpha_i, \beta_i)$ prior for each p_i.
for $t = 1, 2, \ldots, T$ **do**:
  For each arm i, get sample $s_{i,t} \sim Beta(\alpha_i, \beta_i)$. ▹ Each is a float in $[0, 1]$.
  Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} s_{i,t}$. ▹ This "bakes in" exploration!
  Receive reward $r_t \sim Ber(p_{a_t})$.
  if $r_t == 1$ **then** $\alpha_{a_t} \leftarrow \alpha_{a_t} + 1$. ▹ Increment number of "successes".
  else if $r_t == 0$ **then** $\beta_{a_t} \leftarrow \beta_{a_t} + 1$. ▹ Increment number of "failures".

Arm (i)	True p_i	α_i	β_i	$s_{i,t}$
1	0.5	1	1	0.44
2	0.2	1	2	0.27
3	0.9	2	1	0.86

Time (t)	Arm Pulled (a_t)	Reward (r_t)
3	3	0

Observe reward 1 with probability 0.9 and 0 with probability 0.1.

Thompson Example



Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

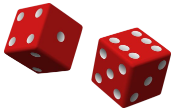
```
For each arm i ∈ {1, ..., K}, initialize α_i = β_i = 1.          ▹ Set Beta(α_i, β_i) prior for each p_i.
for t = 1, 2, ..., T do:
    For each arm i, get sample s_{i,t} ~ Beta(α_i, β_i).          ▹ Each is a float in [0, 1].
    Pull arm a_t = arg max_{i∈{1,2,...,K}} s_{i,t}.               ▹ This "bakes in" exploration!
    Receive reward r_t ~ Ber(p_{a_t}).
    if r_t == 1 then α_{a_t} ← α_{a_t} + 1.                       ▹ Increment number of "successes".
    else if r_t == 0 then β_{a_t} ← β_{a_t} + 1.                  ▹ Increment number of "failures".
```

Arm (i)	True p_i	α_i	β_i	$s_{i,t}$
1	0.5	1	1	0.44
2	0.2	1	2	0.27
3	0.9	2	2	0.86

Time (t)	Arm Pulled (a_t)	Reward (r_t)
3	3	0

Add a count of 1 to the failures :(.

Thompson Example



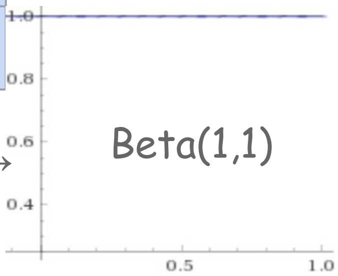
Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

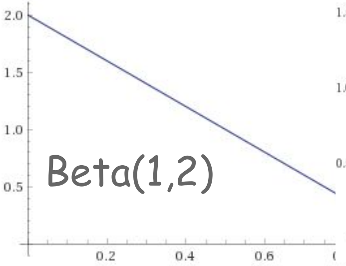
For each arm $i \in \{1, \ldots, K\}$, initialize $\alpha_i = \beta_i = 1$. ▹ Set $Beta(\alpha_i, \beta_i)$ prior for each p_i.
for $t = 1, 2, \ldots, T$ **do**:
    For each arm i, get sample $s_{i,t} \sim Beta(\alpha_i, \beta_i)$. ▹ Each is a float in $[0, 1]$.
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} s_{i,t}$. ▹ This "bakes in" exploration!
    Receive reward $r_t \sim Ber(p_{a_t})$.
    if $r_t == 1$ **then** $\alpha_{a_t} \leftarrow \alpha_{a_t} + 1$. ▹ Increment number of "successes".
    else if $r_t == 0$ **then** $\beta_{a_t} \leftarrow \beta_{a_t} + 1$. ▹ Increment number of "failures".

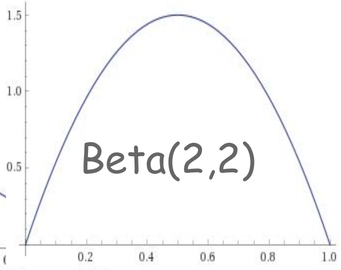
Arm (i)	True p_i	α_i	β_i	$s_{i,t}$
1	0.5	1	1	0.63
2	0.2	1	2	0.15
3	0.9	2	2	0.44

Time (t)	Arm Pulled (a_t)	Reward (r_t)
4		

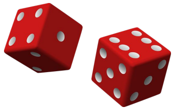
Sample from each arm's Beta distribution →







Thompson Example



Algorithm 5 Thompson Sampling Algorithm for Beta-Bernoulli Bandits

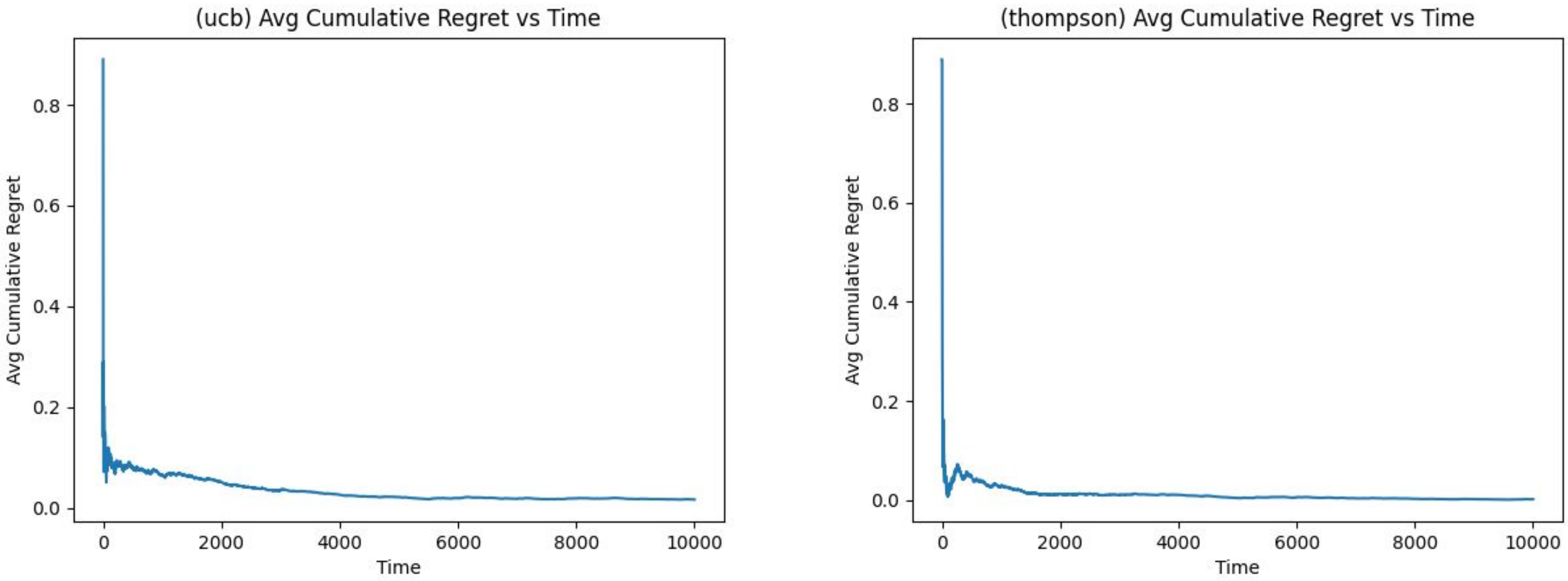
For each arm $i \in \{1, \ldots, K\}$, initialize $\alpha_i = \beta_i = 1$. ▷ Set $Beta(\alpha_i, \beta_i)$ prior for each p_i.
for $t = 1, 2, \ldots, T$ **do**:
    For each arm i, get sample $s_{i,t} \sim Beta(\alpha_i, \beta_i)$. ▷ Each is a float in $[0, 1]$.
    Pull arm $a_t = \arg\max_{i \in \{1,2,\ldots,K\}} s_{i,t}$. ▷ This "bakes in" exploration!
    Receive reward $r_t \sim Ber(p_{a_t})$.
    if $r_t == 1$ **then** $\alpha_{a_t} \leftarrow \alpha_{a_t} + 1$. ▷ Increment number of "successes".
    else if $r_t == 0$ **then** $\beta_{a_t} \leftarrow \beta_{a_t} + 1$. ▷ Increment number of "failures".

Arm (i)	True p_i	α_i	β_i	$s_{i,t}$
1	0.5	1	1	0.63
2	0.2	1	2	0.15
3	0.9	2	2	0.44

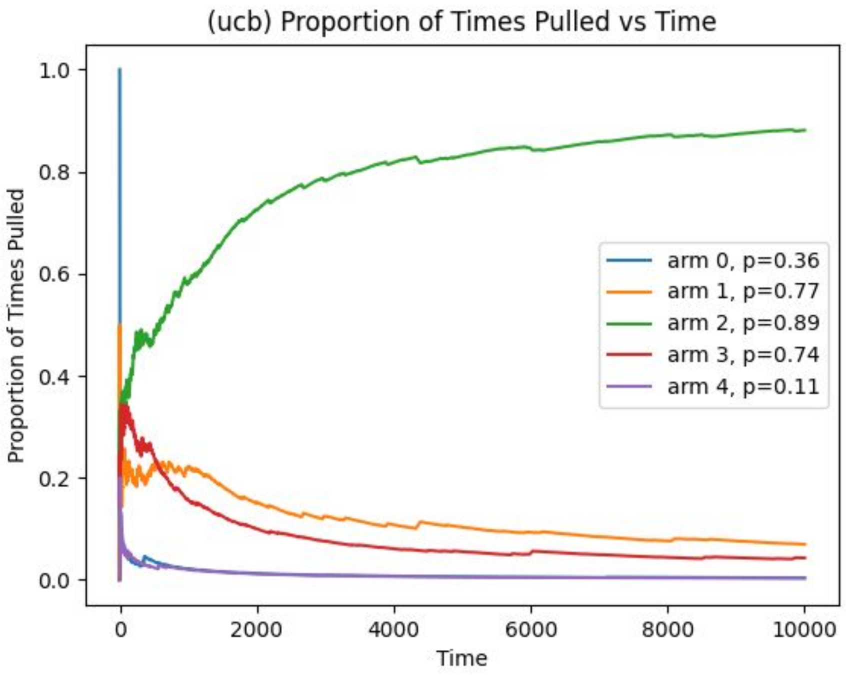
Time (t)	Arm Pulled (a_t)	Reward (r_t)
4		

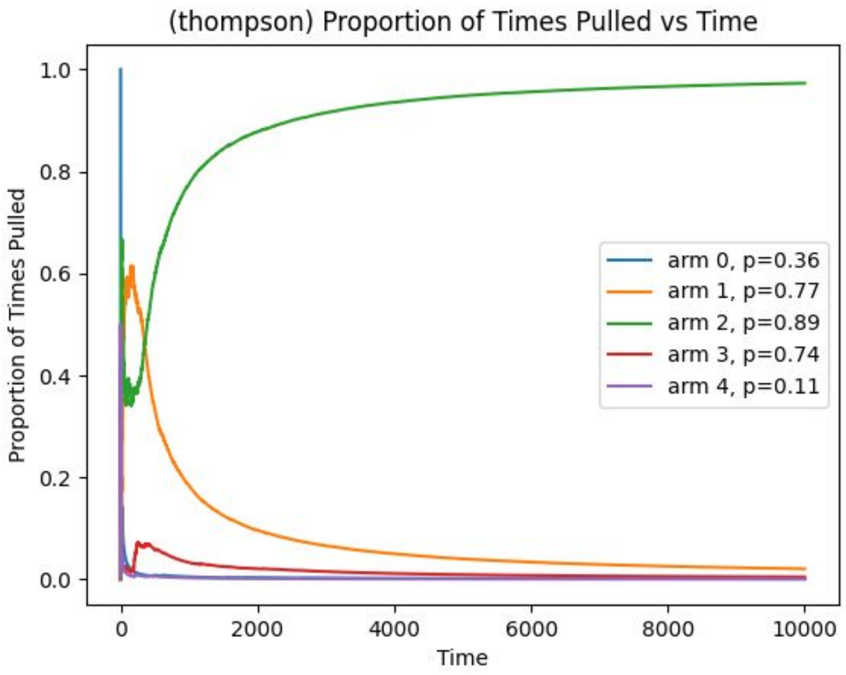
And so on!!! Notice how we explore because there's some chance the "best" arm will have a lower sample occasionally and let other arms win!

UCB Vs Thompson Sampling: Avg Regret over Time



UCB Vs Thompson Sampling: Proportion of Times Pulled





Traditional A/B (Hypothesis) Testing

A large company wants to experiment releasing a new feature/modification.

Assign
- 99% of population to control group (current feature)
- 1% to experimental group (new feature).

Traditional A/B (Hypothesis) Testing

A large company wants to experiment releasing a new feature/modification.

Assign
- 99% of population to control group (current feature)
- 1% to experimental group (new feature).

This has the following consequences:
- If the new feature is “bad”, very small percentage of the population sees it, so company protects itself.

Traditional A/B (Hypothesis) Testing

A large company wants to experiment releasing a new feature/modification.

Assign

- 99% of population to control group (current feature)
- 1% to experimental group (new feature).

This has the following consequences:

- If the new feature is “bad”, very small percentage of the population sees it, so company protects itself.
- If the new feature is “good”, very small percentage of the population sees it, so company may lose revenue.

Traditional A/B (Hypothesis) Testing

A large company wants to experiment releasing a new feature/modification.

Assign
- 99% of population to control group (current feature)
- 1% to experimental group (new feature).

This has the following consequences:
- If the new feature is “bad”, very small percentage of the population sees it, so company protects itself.
- If the new feature is “good”, very small percentage of the population sees it, so company may lose revenue.

Can we do better? Can we adaptively assign subjects to each group based on how each is performing rather than deciding at the beginning?

Traditional A/B (Hypothesis) Testing

A large company

Assign

- 99% of po
- 1% to expe

This has the fo

- If the new … company protects i
- If the new … company may lose r

Can we do bette … n is performing rath

Modern A/B (Hypothesis) Testing

A large company wants to experiment releasing a new feature/modification.

Assign

- Arm 1: Current Feature
- Arm 2: New feature

Modern A/B (Hypothesis) Testing

A large company wants to experiment releasing a new feature/modification.

Assign
- Arm 1: Current Feature
- Arm 2: New feature

When feature is requested by some user, use Multi-Armed Bandit algorithm to decide which feature to show!

Modern A/B (Hypothesis) Testing

A large company wants to experiment releasing a new feature/modification.

Assign

- Arm 1: Current Feature
- Arm 2: New feature

When feature is requested by some user, use Multi-Armed Bandit algorithm to decide which feature to show!

(Can have any number of features/arms)

Modern A/B (Hypothesis) Testing

When to use Traditional A/B Testing:

- Need to collect data for critical business decisions.
- Need statistical confidence in all your results and impact. Want to learn even about treatments that didn't perform well.
- The reward is not immediate (e.g., if drug testing, don't have time to wait for each patient to finish before experimenting with next patient).
- Optimize/measure multiple metrics, not just one.

Modern A/B (Hypothesis) Testing

When to use Traditional A/B Testing:

- Need to collect data for critical business decisions.
- Need statistical confidence in all your results and impact. Want to learn even about treatments that didn't perform well.
- The reward is not immediate (e.g., if drug testing, don't have time to wait for each patient to finish before experimenting with next patient).
- Optimize/measure multiple metrics, not just one.

When to use Multi-Armed Bandits:

- No need for interpreting results, just maximize reward (typically revenue/engagement)
- The opportunity cost is high (if advertising a car, losing a conversion is >=$20,000)
- Can add/remove arms in the middle of an experiment! Cannot do with A/B tests.

Modern A/B (Hypothesis) Testing

When to use Traditional A/B Testing:

- Need to collect data for critical business decisions.
- Need statistical confidence in all your results and impact. Want to learn even about treatments that didn't perform well.
- The reward is not immediate (e.g., if drug testing, don't have time to wait for each patient to finish before experimenting with next patient).
- Optimize/measure multiple metrics, not just one.

When to use Multi-Armed Bandits:

- No need for interpreting results, just maximize reward (typically revenue/engagement)
- The opportunity cost is high (if advertising a car, losing a conversion is >=$20,000)
- Can add/remove arms in the middle of an experiment! Cannot do with A/B tests.

The study of Multi-Armed Bandits can be categorized as:

- Statistics
- Optimization
- "Reinforcement Learning" (subfield of Machine Learning)