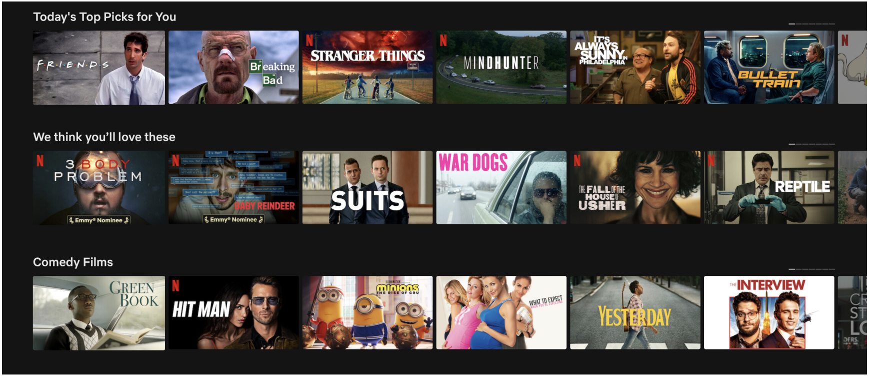
A Bandit in a Bandit
Adaptive Windowing for Non-Stationary Contextual Multi-Armed Bandits

Jack Kerr [1]   Theo Crookes [2]

[1]University of Warwick   [2]Lancaster University

30/8/24

Motivation


Today's Top Picks for You
F·R·I·E·N·D·S
Breaking Bad
STRANGER THINGS
MINDHUNTER
IT'S ALWAYS SUNNY IN PHILADELPHIA
BULLET TRAIN
We think you'll love these
3 BODY PROBLEM
Emmy® Nominee
BABY REINDEER
Emmy® Nominee
SUITS
WAR DOGS
THE FALL OF THE HOUSE OF USHER
REPTILE
Comedy Films
GREEN BOOK
HIT MAN
minions THE RISE OF GRU
WHAT TO EXPECT
YESTERDAY
THE INTERVIEW

The End Goal

With things such as movie recommendations preferences of the users may change over time. Christmas movies are an example of this as we may want to only recommend these seasonally.

Contextual Bandits

- Known time horizon T

Contextual Bandits

- Known time horizon T
- There are N arms

Contextual Bandits

- Known time horizon T
- There are N arms
- Each arm i has some context $\mathbf{b}_i(t) \in \mathbb{R}^d$ attached to it

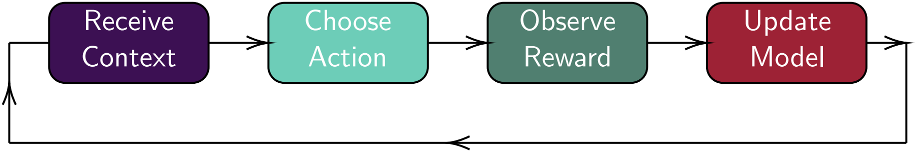
Contextual Bandits

- Known time horizon T
- There are N arms
- Each arm i has some context $\mathbf{b}_i(t) \in \mathbb{R}^d$ attached to it
- $r_i(t)$ is the reward received at time t from arm i

Contextual Bandits

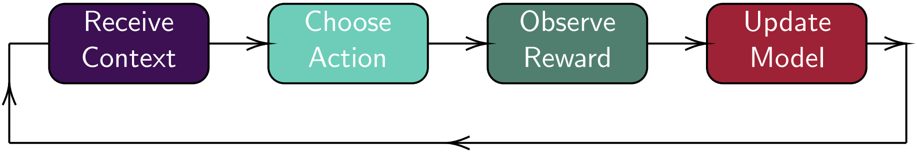
- Known time horizon T
- There are N arms
- Each arm i has some context $\mathbf{b}_i(t) \in \mathbb{R}^d$ attached to it
- $r_i(t)$ is the reward received at time t from arm i
- $\boldsymbol{\mu} \in \mathbb{R}^d$ is the true but unknown parameter such that $\mathbb{E}[r_i(t)|\mathbf{b}_i(t)] = \mathbf{b}_i^T(t)\boldsymbol{\mu}$

Setup



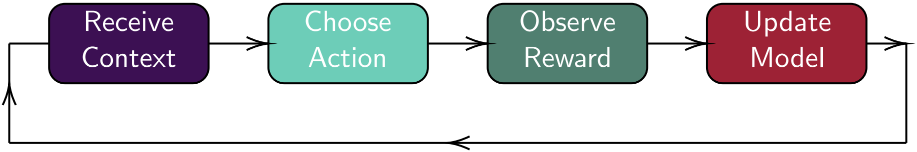
1. Receive context for each arm: $\mathbf{b}_i(t)$

Setup



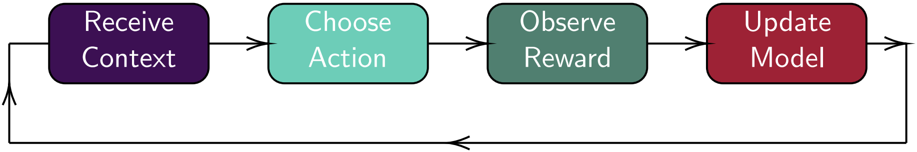
1. Receive context for each arm: $\mathbf{b}_i(t)$
2. Choose an arm to play: a_t

Setup



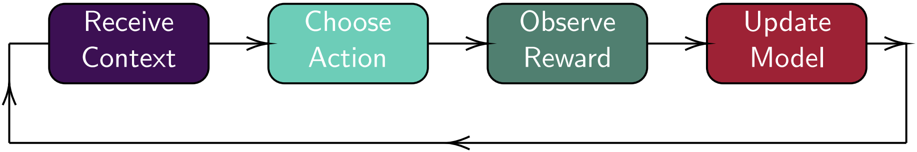
1. Receive context for each arm: $\mathbf{b}_i(t)$
2. Choose an arm to play: a_t
3. Observe the reward: $r_t = \mathbf{b}_{a_t}(t)^T \boldsymbol{\mu} + \epsilon_{a_t,t}$

Setup



1. Receive context for each arm: $\mathbf{b}_i(t)$
2. Choose an arm to play: a_t
3. Observe the reward: $r_t = \mathbf{b}_{a_t}(t)^T \boldsymbol{\mu} + \epsilon_{a_t,t}$
4. Update model parameters

Setup



1. Receive context for each arm: $\mathbf{b}_i(t)$
2. Choose an arm to play: a_t
3. Observe the reward: $r_t = \mathbf{b}_{a_t}(t)^T \boldsymbol{\mu} + \epsilon_{a_t,t}$
4. Update model parameters
5. Repeat

Upper Confidence Bound

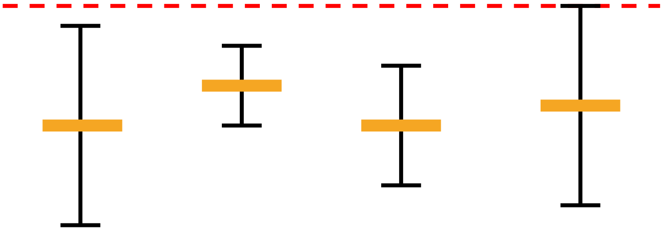
At each time t, we compute the UCB value for each arm i and then choose the arm with the highest UCB value.

$$UCB_i(t) = \text{Estimated Reward} + \text{Uncertainty}$$

Upper Confidence Bound

At each time t, we compute the UCB value for each arm i and then choose the arm with the highest UCB value.

$$UCB_i(t) = \text{Estimated Reward} + \text{Uncertainty}$$



Upper Confidence Bound

At each time t, we compute the UCB value for each arm i and then choose the arm with the highest UCB value.

$$UCB_i(t) = \mathbf{b}_i^T(t)\hat{\boldsymbol{\mu}}(t) + \alpha\sqrt{\mathbf{b}_i^T(t)B(t)^{-1}\mathbf{b}_i(t)}$$

Where,

$$B(t) = \lambda \mathbf{I}_d + \sum_{k=1}^{t-1} \mathbf{b}_{a_k}(k)\mathbf{b}_{a_k}^T(k)$$

$$\hat{\boldsymbol{\mu}}(t) = B(t)^{-1} \sum_{k=1}^{t-1} \mathbf{b}_{a_k}(k) r_k$$

Regret

Regret is a way to measure the performance of contextual multi-armed bandit algorithms where the goal is to minimise the total regret over the time horizon T.

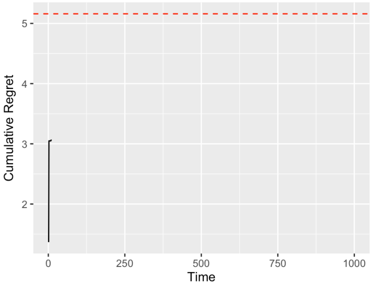
Regret

Regret is a way to measure the performance of contextual multi-armed bandit algorithms where the goal is to minimise the total regret over the time horizon T.

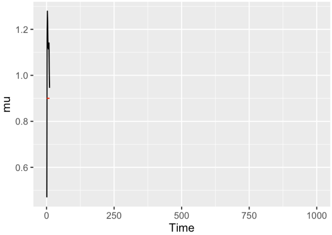
$$\mathcal{R}(T) = \sum_{t=1}^{T} \mathbf{b}_{a_t^*}^T(t)\,\boldsymbol{\mu} - \mathbf{b}_{a_t}^T(t)\,\boldsymbol{\mu}$$

- a_t^* represents the best arm at time t
- a_t represents the arm chosen at time t

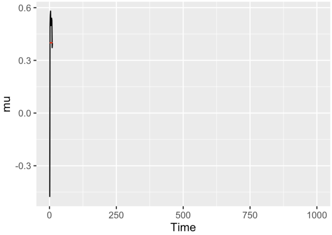
Example Iteration



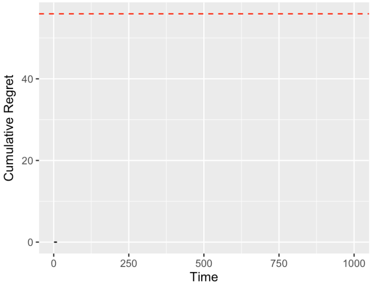
μ_1 Values

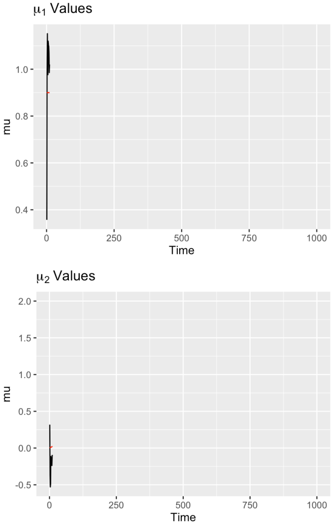


μ_2 Values



Example Iteration





Non-Stationary

We now focus on the case where $\boldsymbol{\mu}(t)$ is a function of time. As in the last example:

$$\boldsymbol{\mu}(t) = (\mu_1(t), \mu_2(t))$$
$$\mu_1(t) = 0.9$$
$$\mu_2(t) = 2\frac{t}{T}$$

Non-Stationary

We now focus on the case where $\boldsymbol{\mu}(t)$ is a function of time. As in the last example:

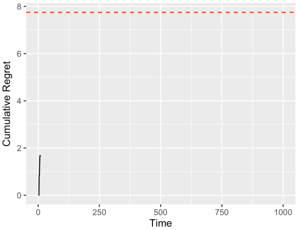
$$\boldsymbol{\mu}(t) = (\mu_1(t), \mu_2(t))$$
$$\mu_1(t) = 0.9$$
$$\mu_2(t) = 2\frac{t}{T}$$

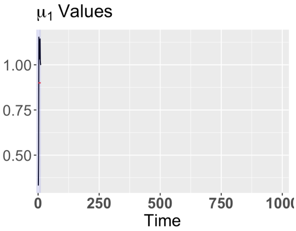
We have 2 main ways to deal with these cases,

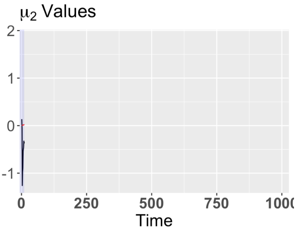
1. Discounting
2. Sliding Window

Sliding Window

Sliding window methods involve only using information from at most τ time steps ago to estimate $\mu(t)$.







Sliding Window Algorithm

The SW-UCB algorithm is very similar to the standard UCB algorithm with some slight changes to the estimates of $\hat{\boldsymbol{\mu}}(t)$ and B_t.
We instead have:

$$UCB_i(t) = \mathbf{b}_i^T(t)\hat{\boldsymbol{\mu}}_\tau(t) + \alpha\sqrt{\mathbf{b}_i^T(t)B_\tau^{-1}(t)\mathbf{b}_i(t)}$$

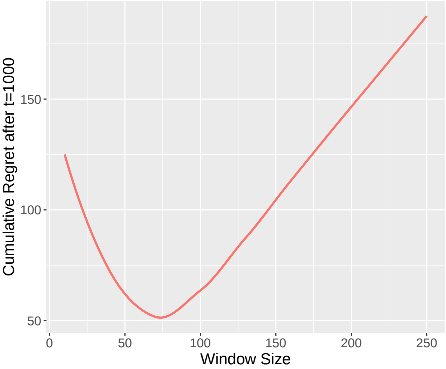
Where,

$$B_\tau(t) = \lambda \mathbf{I}_d + \sum_{k=max(1,t-\tau)}^{t-1} \mathbf{b}_{a_k}(k)\mathbf{b}_{a_k}^T(k)$$

$$\hat{\boldsymbol{\mu}}_\tau(t) = B_t^{-1} \sum_{k=max(1,t-\tau)}^{t-1} \mathbf{b}_{a_k}(k) r_k$$

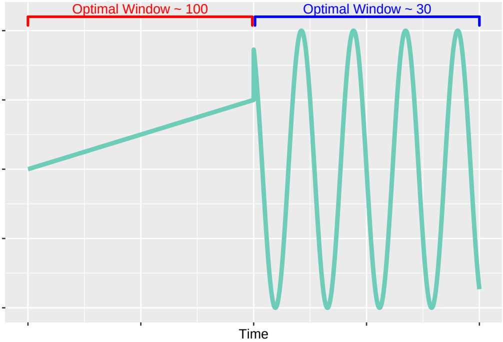
The First Issue

The first problem with these 2 methods is that you need prior knowledge of the behaviour of $\mu(t)$ to pick the optimal window size.



The Second Issue

The second problem with sliding window methods is that they cannot change their window size as time goes on to adapt to a new environment.



Adaptive Window Sizing

We now want to be able to adaptively choose window sizes while we use a SW-UCB algorithm to select which arm to play.

Adaptive Window Sizing

We now want to be able to adaptively choose window sizes while we use a SW-UCB algorithm to select which arm to play.

- Define a maximum window size M

Adaptive Window Sizing

We now want to be able to adaptively choose window sizes while we use a SW-UCB algorithm to select which arm to play.

- Define a maximum window size M
- Define the set of possible window sizes $\mathcal{M} = \{10, 20, \dots, M-10, M\}$

Adaptive Window Sizing

We now want to be able to adaptively choose window sizes while we use a SW-UCB algorithm to select which arm to play.

- Define a maximum window size M
- Define the set of possible window sizes $\mathcal{M} = \{10, 20, \ldots, M - 10, M\}$

We then implement 2 different ways to pick a window size at each time point and assess how good it performs in relation to other window sizes.

Hedging

Hedging relies on each window size having some probability of being sampled and then updating this probability based on its performance.

Hedging

Hedging relies on each window size having some probability of being sampled and then updating this probability based on its performance.

Set $W_1 = \mathbf{1} \in \mathbb{R}^N, \mathbf{x}_1 = \frac{1}{N} W_1$
Play according to UCB for the first M iterations
for all $t = M+1, \ldots, T$ **do**
  Sample $\tau \sim \{10, 20, 30, \ldots, M\}, \quad \mathbb{P}(\tau = i) = \mathbf{x}_t$
  Observe the reward r_t from playing according to SW-UCB with window size τ
  Compute the loss given by some function $\mathbf{g}(r_t, r_{t-1})$
  Update Weights $W_t(\tau) = W_{t-1}(\tau) e^{\mathbf{g}(r_t, r_{t-1})}$
  Set $\mathbf{x}_t = \frac{W_t}{\sum_j W_t(j)}$
end for

$\epsilon-$Greedy (ish)

This method involves using a $\epsilon-$Greedy (ish) Bandit inside of the SW-UCB algorithm to choose the window size.

$\epsilon-$Greedy (ish)

This method involves using a $\epsilon-$Greedy (ish) Bandit inside of the SW-UCB algorithm to choose the window size.

1: Set $\beta_i = 0, P_{i,t} = 0 \; \forall i \in \{10, 20, \ldots, M\}$
2: Play according to UCB for the first M iterations
3: **for all** $t = M + 1, \ldots, T$ **do**
4: Either choose a random window size τ w.p ϵ or choose the window size with the largest β value.
5: Observe the reward r_t from playing according to SW-UCB with window size τ
6: Update the number of times τ has been used
7: Update β_τ
8: **end for**

Update Rule

$$\beta_\tau = \beta_\tau \left(1 - \frac{1}{P_{\tau,t}}\right) + \frac{\left((r_t - r_{t-1}) - \frac{1}{|S_{\tau,t}|}\sum_{s \in S_{\tau,t}}(r_s - r_{s-1})\right)}{P_{\tau,t}}$$

Where,

$$P_{\tau,t} = \{\# \text{ of times } \tau \text{ has been used up to time t}\}$$
$$S_{\tau,t} = \{i \in \{1,\dots,t\} \mid w_i \neq \tau\}$$

Mass Update

One way to speed up the learning process is to update each window size that would have also chosen the arm the chosen window size did with a certain weighting.

Mass Update

One way to speed up the learning process is to update each window size that would have also chosen the arm the chosen window size did with a certain weighting.

- A constant proportion of the reward.

Mass Update

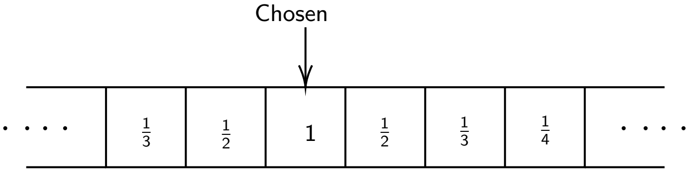
One way to speed up the learning process is to update each window size that would have also chosen the arm the chosen window size did with a certain weighting.

- A constant proportion of the reward.
- If n other window sizes would have selected arm a_t give each of them arms $1/n$ of the reward.

Mass Update

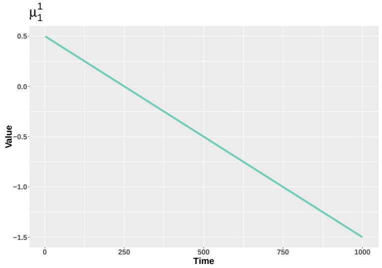
One way to speed up the learning process is to update each window size that would have also chosen the arm the chosen window size did with a certain weighting.

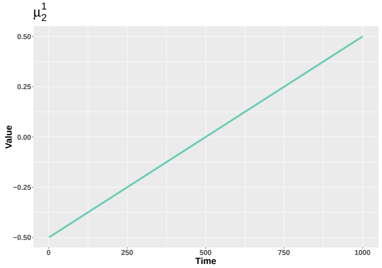
- A constant proportion of the reward.
- If n other window sizes would have selected arm a_t give each of them arms $1/n$ of the reward.
- Define a metric to represent the distance (d) each window is from the chosen one and give them $1/d$ of the reward.



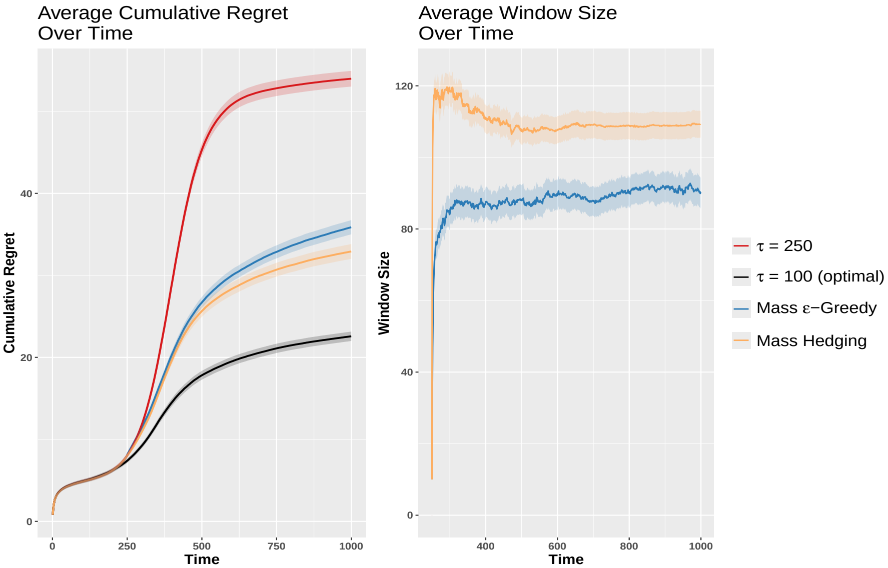
Linear

$$\boldsymbol{\mu}^1(t) = \left(\mu_1^1(t), \mu_2^1(t)\right)$$
$$\mu_1^1(t) = 0.5 - 2\frac{t}{T}$$
$$\mu_2^1(t) = -0.5 + \frac{t}{T}$$

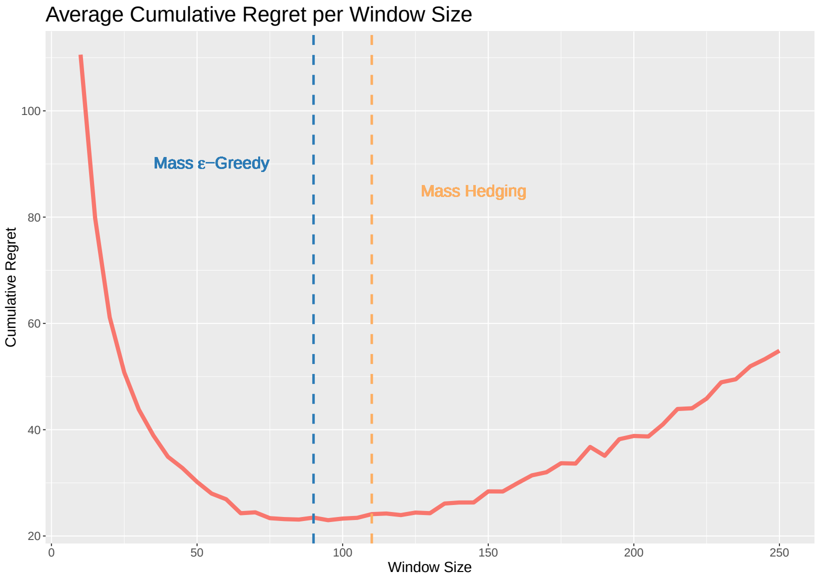




Linear


Average Cumulative Regret
Over Time
Cumulative Regret
Time
0
20
40
0
250
500
750
1000
Average Window Size
Over Time
Window Size
Time
0
40
80
120
400
600
800
1000
τ = 250
τ = 100 (optimal)
Mass ε−Greedy
Mass Hedging

Linear



Average Cumulative Regret per Window Size
Mass ε−Greedy
Mass Hedging
Cumulative Regret
Window Size
20
40
60
80
100
0
50
100
150
200
250

Gi-Soo Kim, Young Suh Hong, Tae Hoon Lee, Myunghee Cho Paik, and Hongsoo Kim.
Bandit-supported care planning for older people with complex health and care needs, 2023.

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

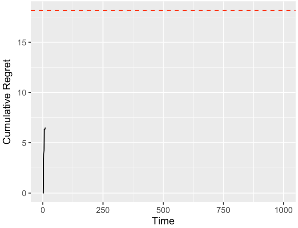
Any Questions?

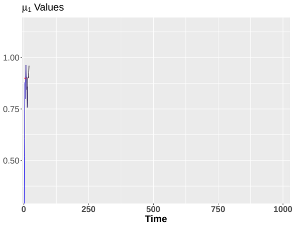
Any Questions?

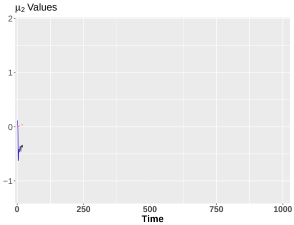
James?

Discounting

The discounting method multiplies all past information by some constant λ to reduce their impact on the current estimate for $\mu(t)$ and place a higher weight on more recent information.



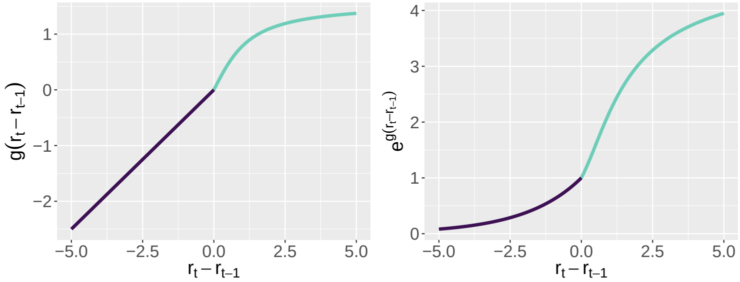




*

Loss Functions

$$\mathbf{g}(r_t, r_{t-1}) = \begin{cases} \arctan(r_t - r_{t-1}) & \textit{if} \quad r_t - r_{t-1} \geq 0 \\ \frac{1}{2}(r_t - r_{t-1}) & \textit{otherwise} \end{cases}$$

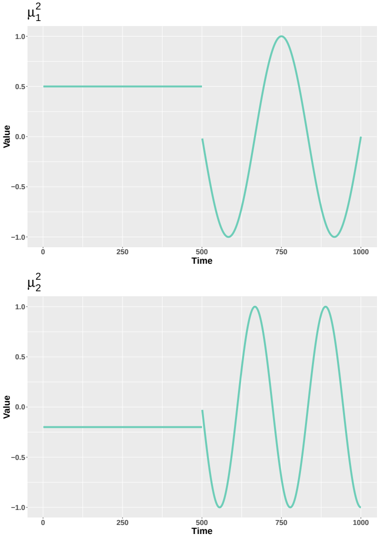


Changing

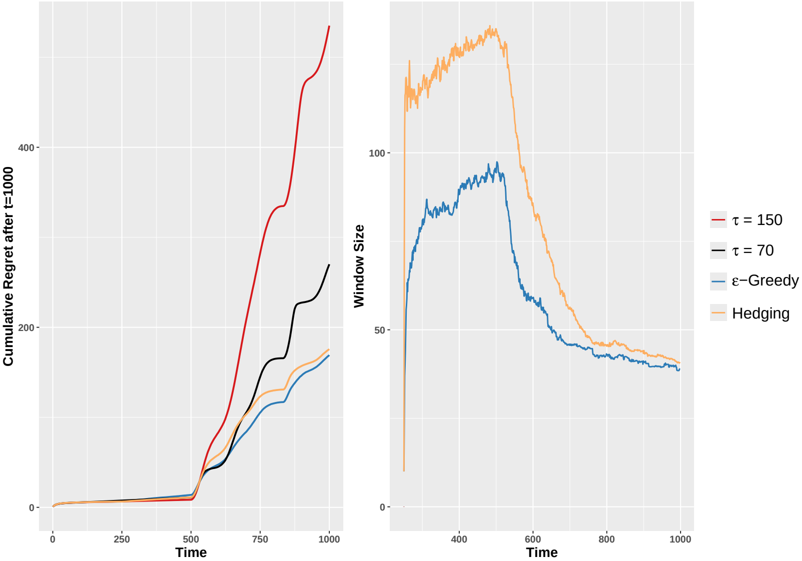
$$\boldsymbol{\mu}^2(t) = \left(\mu_1^2(t), \mu_2^2(t)\right)$$

$$\mu_1^2(t) = \begin{cases} 0.5 & \textit{if} \quad t \leq \frac{1}{2}T \\ \sin\left(6\pi\frac{t}{T}\right) & \text{Otherwise} \end{cases}$$

$$\mu_2^2(t) = \begin{cases} -0.2 & \textit{if} \quad t \leq \frac{1}{2}T \\ \cos\left(9\pi\frac{t}{T}\right) & \text{Otherwise} \end{cases}$$



Changing


Cumulative Regret after t=1000
400
200
0
0
250
500
750
1000
Time
Window Size
100
50
0
400
600
800
1000
Time
τ = 150
τ = 70
ε−Greedy
Hedging

Algorithm 1 UCB

Input: Time horizon T, regularization parameter λ, exploration parameter α
Initialisation: $B = \lambda \mathbf{I}_d$, $\hat{\mu} = \mathbf{0}_d$, $\mathbf{f} = \mathbf{0}_d$

for $t = 1, 2, \ldots, T$ **do**
  for all $a \in \mathcal{A}$ **do**
    $p_a(t) = \mathbf{b}_a^T(t)\hat{\boldsymbol{\mu}}(t) + \alpha\sqrt{\mathbf{b}_a^T(t)B_t^{-1}\mathbf{b}_a^T(t)}$
  end for
  Choose arm $a_t = \arg\max_{a \in \mathcal{A}} p_a(t)$and observe a real-valued payoff r_t
  Update $B = B + \mathbf{b}_{a_t}(t)\mathbf{b}_{a_t}^T(t)$
    $\mathbf{f} = \mathbf{f} + \mathbf{b}_{a_t}(t) r_t$
    $\hat{\boldsymbol{\mu}} = B^{-1}\mathbf{f}$
end for

Algorithm 2 Discounted UCB

Input: Time horizon T, regularization parameter λ, exploration parameter α, discount rate η

Initialisation: $B = \lambda \mathbf{I}_d$, $\hat{\mu} = \mathbf{0}_d$, $\mathbf{f} = \mathbf{0}_d$

for $t = 1, 2, \ldots, T$ **do**
    for all $a \in \mathcal{A}$ **do**
        $p_a(t) = \mathbf{b}_a^T(t)\hat{\boldsymbol{\mu}}(t) + \alpha\sqrt{\mathbf{b}_a^T(t)B_t^{-1}\mathbf{b}_a^T(t)}$
    end for
    Choose arm $a_t = \arg\max_{a \in \mathcal{A}} p_a(t)$and observe a real-valued payoff r_t
    Update $B = \eta B + \mathbf{b}_{a_t}(t)\mathbf{b}_{a_t}^T(t)$
        $\mathbf{f} = \eta \mathbf{f} + \mathbf{b}_{a_t}(t) r_t$
        $\hat{\boldsymbol{\mu}} = B^{-1}\mathbf{f}$
end for

Algorithm 3 SW-UCB

Input: Time horizon T, regularization parameter λ, exploration parameter α

Initialisation: $B = \lambda \mathbf{I}_d$, $\hat{\mu} = \mathbf{0}_d$, $\mathbf{f} = \mathbf{0}_d$

for $t = 1, 2, \ldots, T$ **do**
    for all $a \in \mathcal{A}$ **do**
        $p_a(t) = \mathbf{b}_a^T(t)\hat{\boldsymbol{\mu}}(t) + \alpha\sqrt{\mathbf{b}_a^T(t) B_{t,\tau}^{-1} \mathbf{b}_a^T(t)}$
    end for
    Choose arm $a_t = \arg\max_{a \in \mathcal{A}} p_a(t)$and observe a real-valued payoff r_t
    Calculate $B_{t+1,\tau}$ and $\hat{\boldsymbol{\mu}}_\tau(t+1)$
end for

$$B_{t,\tau} = \lambda \mathbf{I}_d \; + \sum_{k=max(1,t-\tau)}^{t-1} \mathbf{b}_{a_k}(k)\mathbf{b}_{a_k}^T(k)$$

$$\hat{\boldsymbol{\mu}}_\tau(t) = B_{t,\tau}^{-1} \sum_{k=max(1,t-\tau)}^{t-1} \mathbf{b}_{a_k}(k) r_k$$

Algorithm 4 Hedged Sliding Window UCB

Input: Time horizon T, regularization parameter λ, exploration parameter α, max window size M

Initialisation: $B_1 = \lambda \mathbf{I}_d$, $W_1 = \mathbf{1} \in \mathbb{R}^N, \mathbf{x}_1 = \frac{1}{N} W_1$, $\hat{\mu} = \mathbf{0}_d$, $\tau = M$

for $t = 1, 2, \ldots, M$ **do**
    for all $a \in \mathcal{A}$ **do**
        $p_a(t) = \mathbf{b}_a^T(t)\hat{\boldsymbol{\mu}}_\tau(t) + \alpha\sqrt{\mathbf{b}_a^T(t)\mathbf{B}_{t,\tau}^{-1}\mathbf{b}_a^T(t)}$
    end for
    Choose arm $a_t = \arg\max_{a \in \mathcal{A}} p_a(t)$and observe a real-valued payoff r_t
    Compute $B_{t+1,\tau}$ and $\hat{\boldsymbol{\mu}}_\tau(t+1)$
end for
Sample $\tau \sim \{10, 20, 30, \ldots, M\}$, $\quad \mathbb{P}(\tau = i) = \mathbf{x}_1$
Compute $B_{M+1,\tau}$ and $\hat{\boldsymbol{\mu}}_\tau(M+1)$
for $t = M+1, M+2, \ldots, T$ **do**
    for all $a \in \mathcal{A}$ **do**
        $p_a(t) = \mathbf{b}_a^T(t)\hat{\boldsymbol{\mu}}_\tau(t) + \alpha\sqrt{\mathbf{b}_a^T(t)\mathbf{B}_{t,\tau}^{-1}\mathbf{b}_a^T(t)}$
    end for
    Choose arm $a_t = \arg\max_{a \in \mathcal{A}} p_a(t)$and observe a real-valued payoff r_t
    Observe reward $g(r_t, r_{t-1})$
    Update Weights $W_t(\tau) = W_{t-1}(\tau)e^{g(r_t, r_{t-1})}$
    Set $\mathbf{x}_t = \frac{W_t}{\sum_j W_t(j)}$
    Sample $\tau \sim \{10, 20, 30, \ldots, M\}$, $\quad \mathbb{P}(\tau = i) = \mathbf{x}_t$
    Compute $B_{t+1,\tau}$ and $\hat{\boldsymbol{\mu}}_\tau(t+1)$
end for

Algorithm 5 ϵ-Greedy Sliding Window UCB

Input: Time horizon T, regularization parameter λ, exploration parameter α, exploration rate ϵ, max window size M

Initialisation: $B(1) = \lambda \mathbf{I}_d$, $\hat{\mu}(1) = \mathbf{0}_d$, $\tau = M$, $P_i = 0, \beta_i = 0 \; \forall i$

for $t = 1, 2, \ldots, M$ **do**

  for all $a \in \mathcal{A}$ **do**

    $p_a(t) = \mathbf{b}_a^T(t)\hat{\boldsymbol{\mu}}_\tau(t) + \alpha\sqrt{\mathbf{b}_a^T(t)\mathbf{B}_{t,\tau}^{-1}\mathbf{b}_a^T(t)}$

  end for

  Choose arm $a_t = \arg\max_{a \in \mathcal{A}} p_a(t)$and observe a real-valued payoff r_t

  Compute $B_{t+1,\tau}$ and $\hat{\boldsymbol{\mu}}_\tau(t+1)$

end for

Set

$$\tau = \begin{cases} \text{Random size} & \text{w.p} \quad \epsilon \\ \arg\max_{a \in \mathcal{A}} \beta_a & \text{w.p} \quad 1 - \epsilon \end{cases}$$

Compute $B_{M+1,\tau}$ and $\hat{\boldsymbol{\mu}}_\tau(M+1)$

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

  for all $a \in \mathcal{A}$ **do**

    $p_a(t) = \mathbf{b}_a^T(t)\hat{\boldsymbol{\mu}}_\tau(t) + \alpha\sqrt{\mathbf{b}_a^T(t)\mathbf{B}_{t,\tau}^{-1}\mathbf{b}_a^T(t)}$

  end for

  Choose arm $a_t = \arg\max_{a \in \mathcal{A}} p_a(t)$and observe a real-valued payoff r_t

  Update $P_\tau = P_\tau + 1$

  Update $\beta_\tau \; = \beta_\tau \left(1 - \frac{1}{P_\tau}\right) \; + \; \frac{\left((r_t - r_{t-1}) - \frac{1}{|S_{\tau,t}|}\sum_{s \in S_{\tau,t}}(r_s - r_{s-1})\right)}{P_\tau}$

  Set

$$\tau = \begin{cases} \text{Random size} & \text{w.p} \quad \epsilon \\ \arg\max_{a \in \mathcal{A}} \beta(a) & \text{w.p} \quad 1 - \epsilon \end{cases}$$

  Compute $B_{t+1,\tau}$ and $\hat{\boldsymbol{\mu}}_\tau(t+1)$

end for