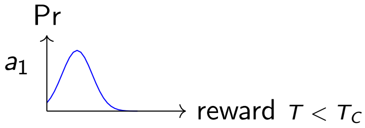
Online Learning Applications

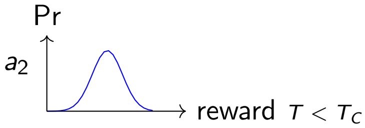
Part 11: Non-stationary environments

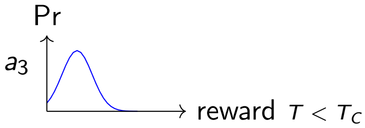
Introduction to non-stationary environments

Non-stationary environments

Rewards change over time

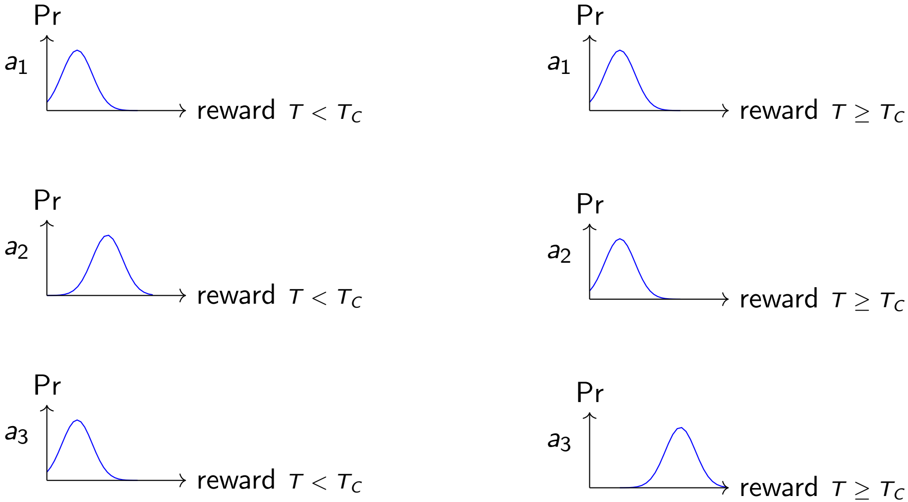






Non-stationary environments

Rewards change over time



Unrestricted non-stationary environments

Without any restriction, a non-stationary environment is equivalent to an adversarial environment.

- We have seen algorithms for adversarial environments
- These algorithms have worse performances than algorithms for stochastic environments

Can we do better when the environment is "slightly" non-stationary?

We consider two types of non-stationary environments:

- With abrupt changes
- With smooth changes

Abrupt changes

Abrupt changes

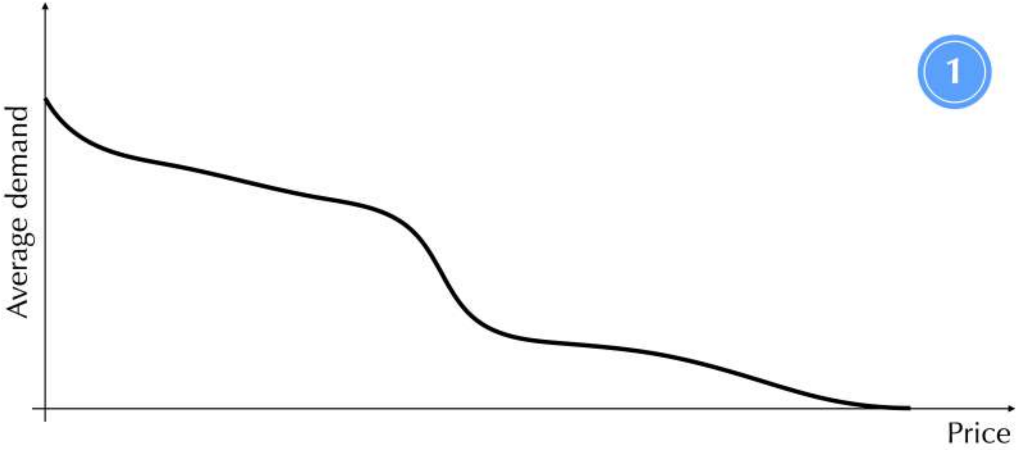
The reward function changes **abruptly** a few times:

- The time horizon is divided in **phases**
- In every phase the reward function is constant

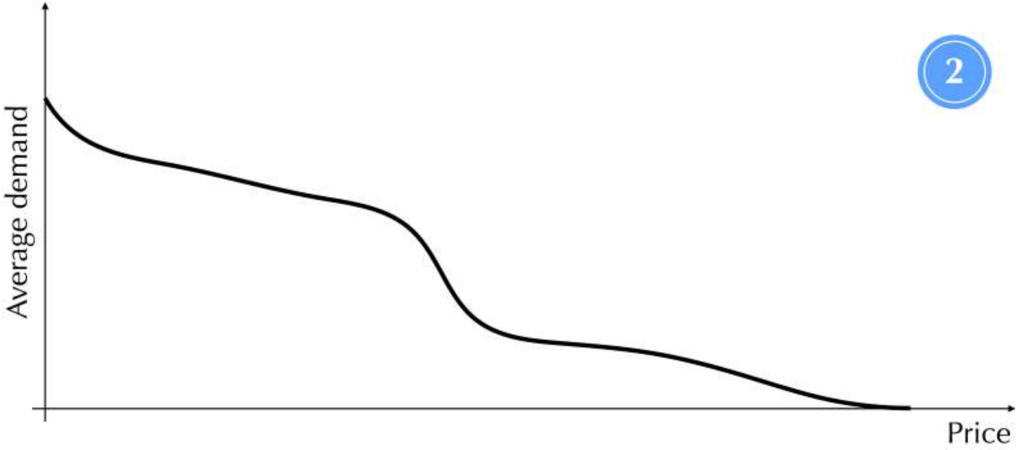
Some example of abrupt changes are:

- A new product enters the market (pricing)
- A new bidder starts to participate to auctions (advertising)
- The preferences of the users change (matching)

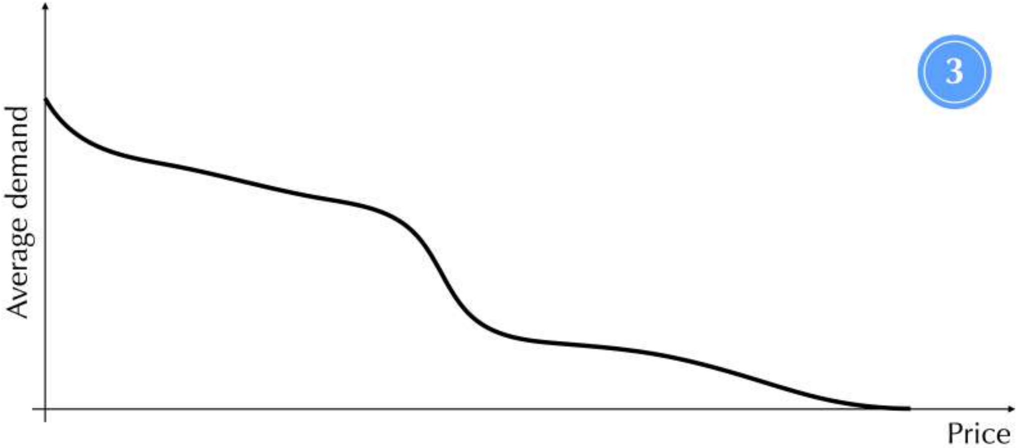
Example of abrupt changes



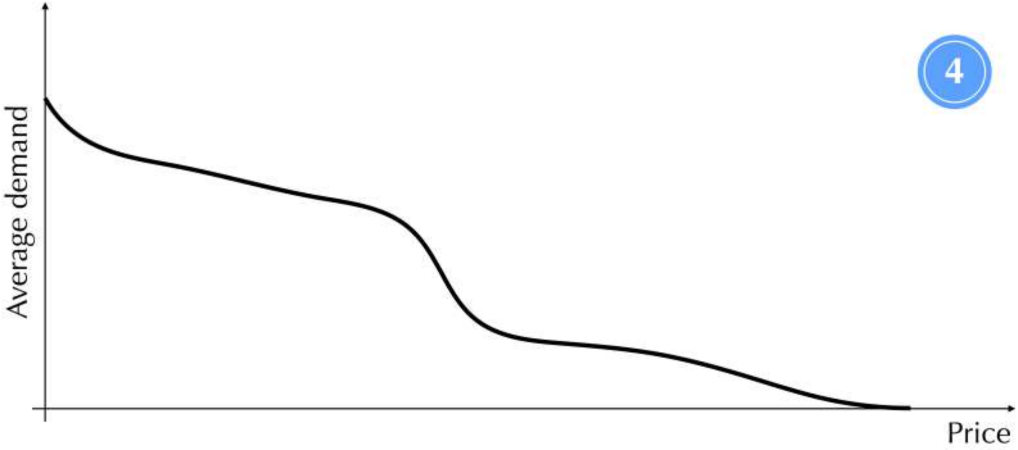
Example of abrupt changes



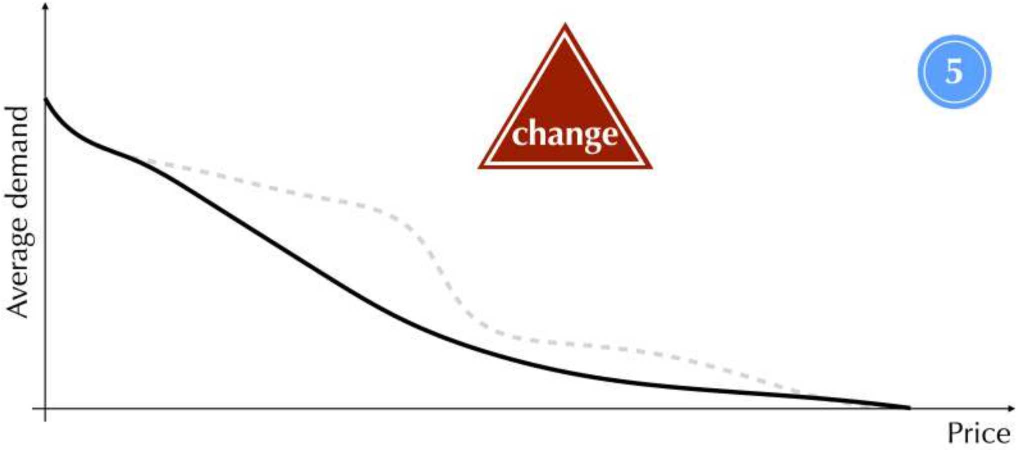
Example of abrupt changes



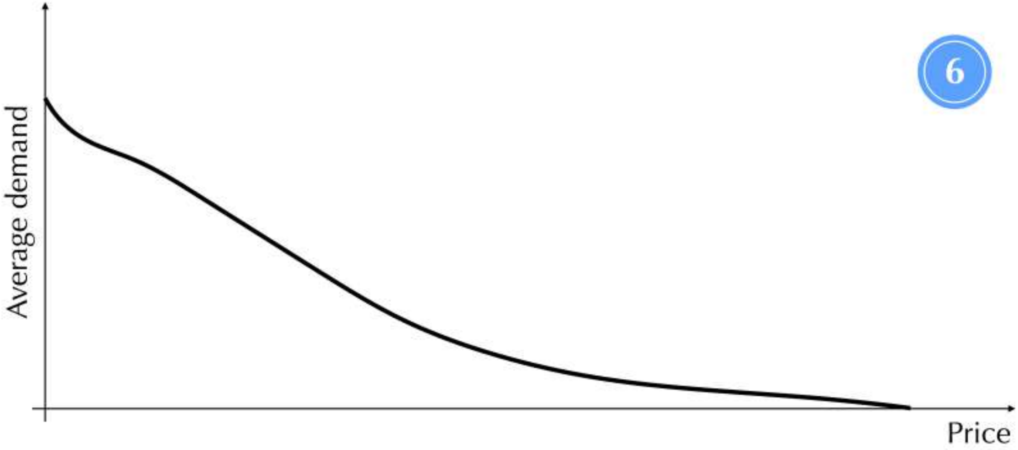
Example of abrupt changes



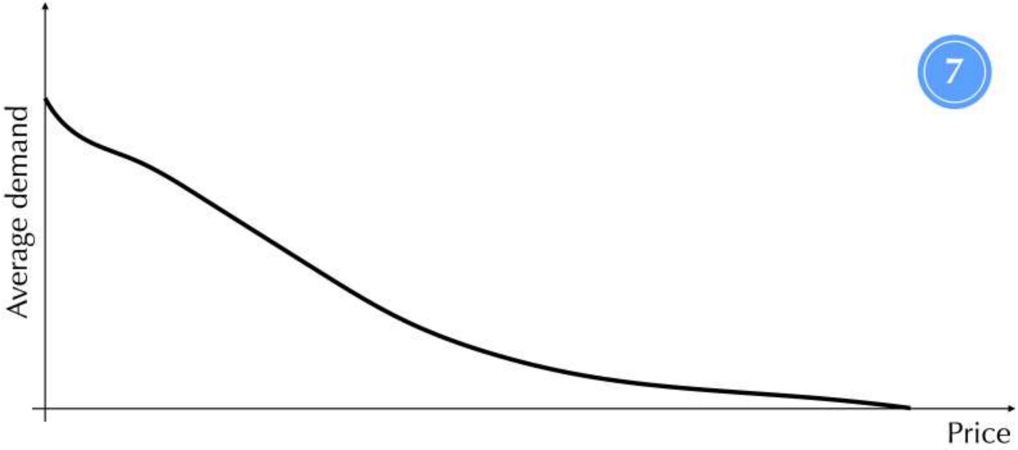
Example of abrupt changes



Example of abrupt changes



Example of abrupt changes



Example of abrupt changes

Smooth changes

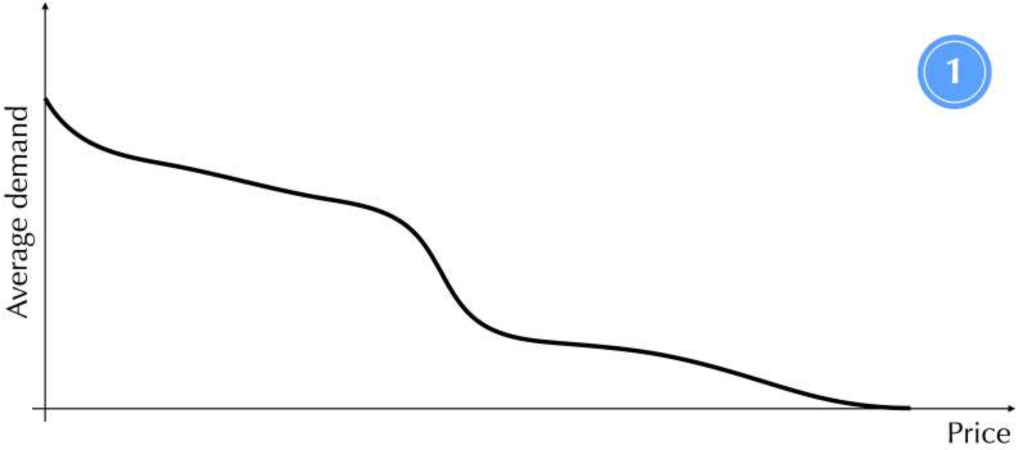
The reward function changes **slowly**:

- The reward function changes a bit at each round
- The reward function changes **continuously** during time

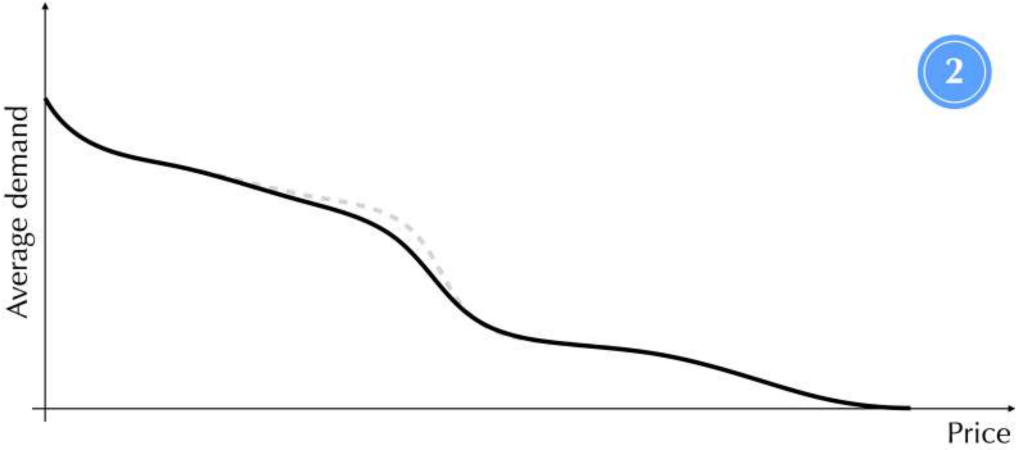
Some examples of smooth changes:

- A product that looses attractiveness during time (pricing)
- The fluctuation in the bidders participating to auctions (advertising)
- The preferences of the users that change slowly over time (matching)

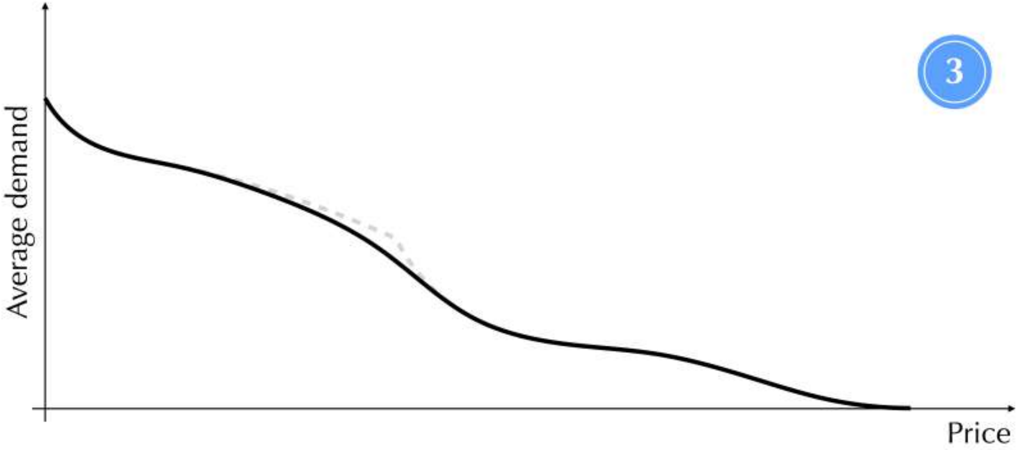
Example of smooth changes



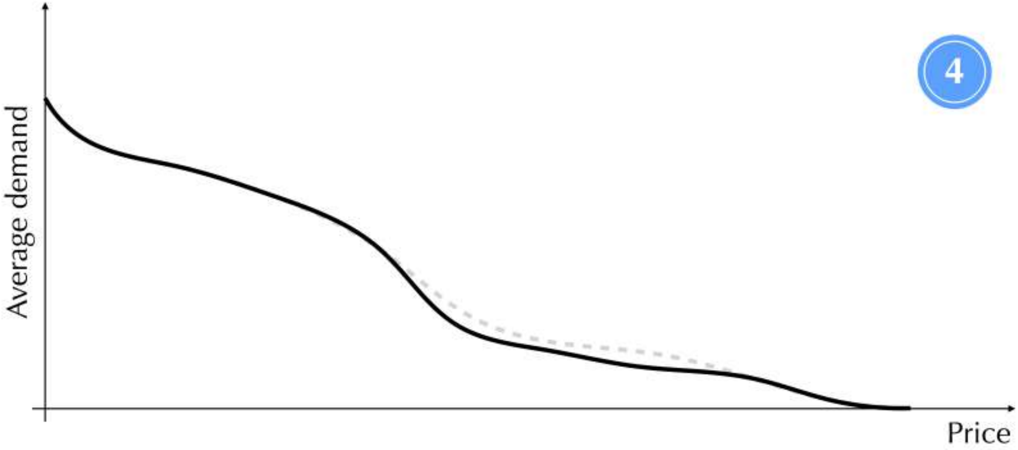
Example of smooth changes



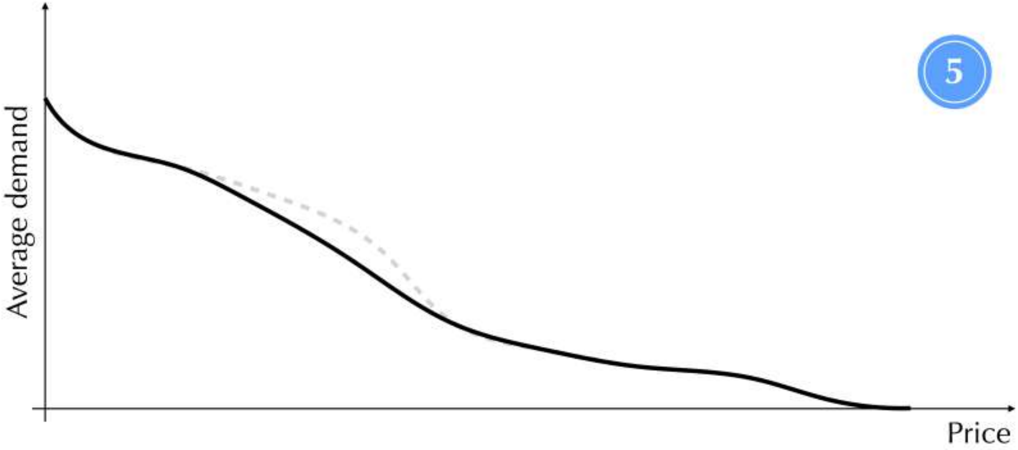
Example of smooth changes



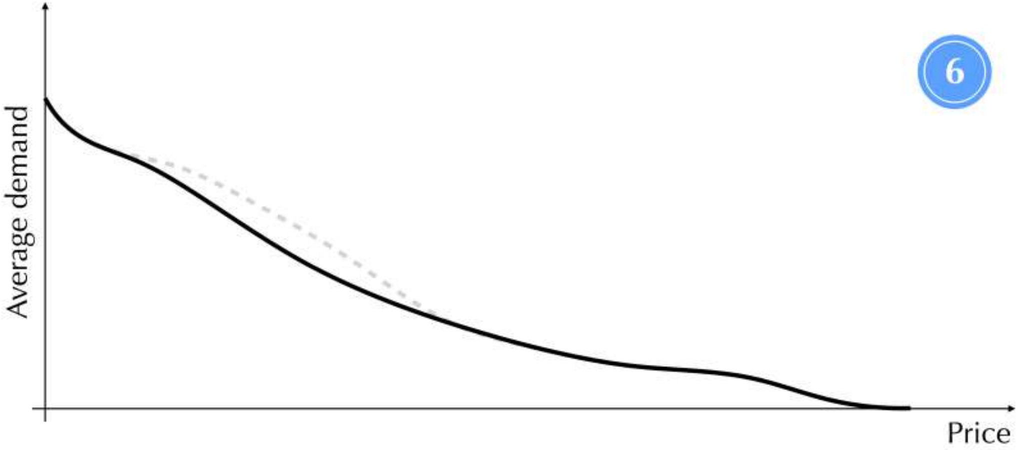
Example of smooth changes



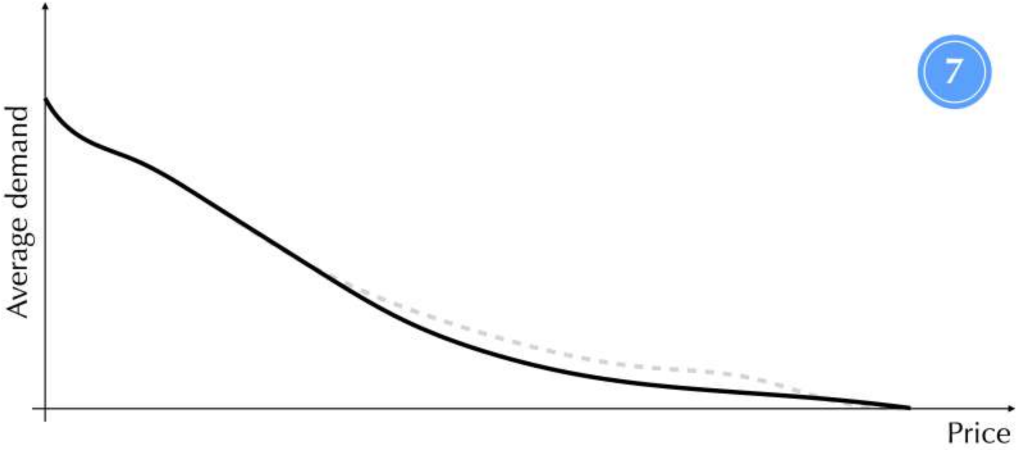
Example of smooth changes



Example of smooth changes



Example of smooth changes



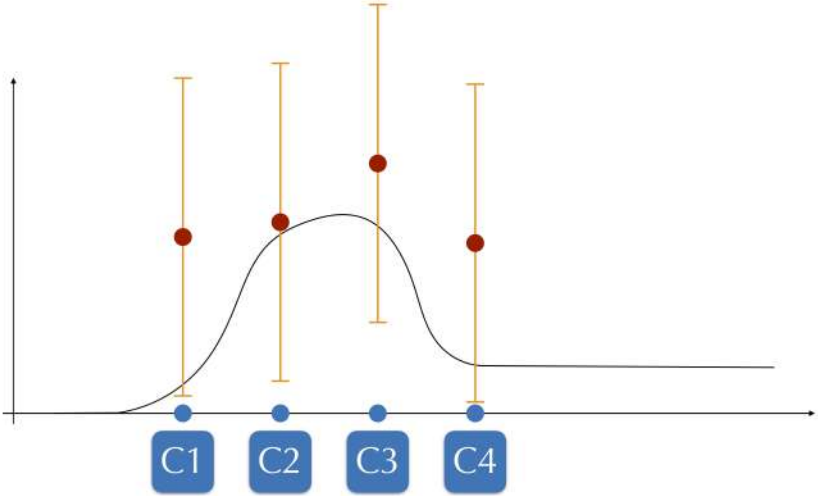
Failure of algorithms for stochastic environments

Failure of standard algorithms

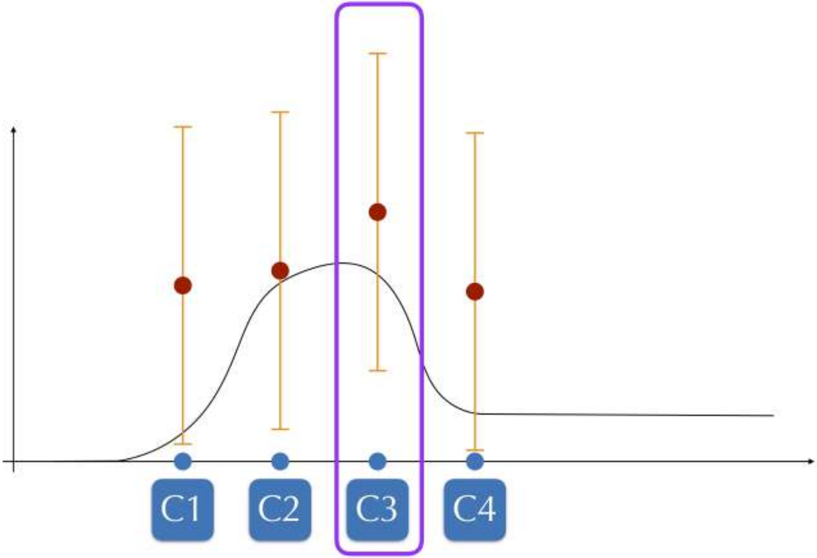
Algorithms for stochastic environments **fail** in non-stochastic environments.

- These algorithms reduce exploration over time and "converge" to an optimal arm
- UCB1 reduces confidence bounds over time
- TS converges to a belief on the reward function

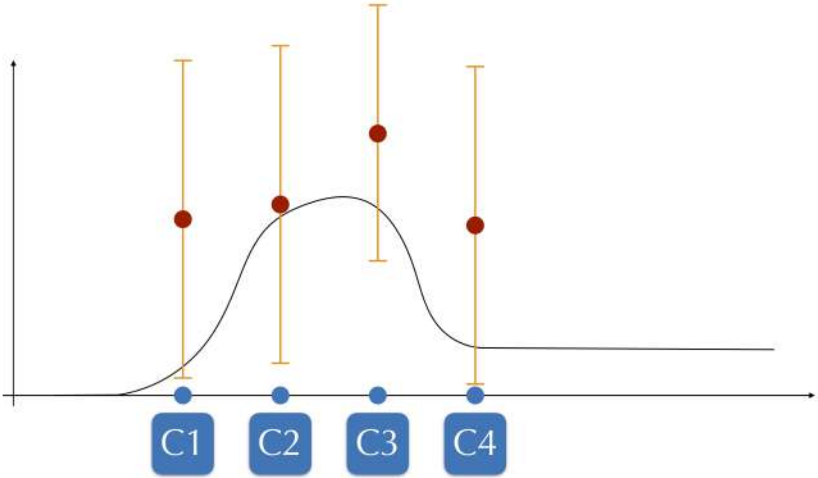
Failure of UCB1



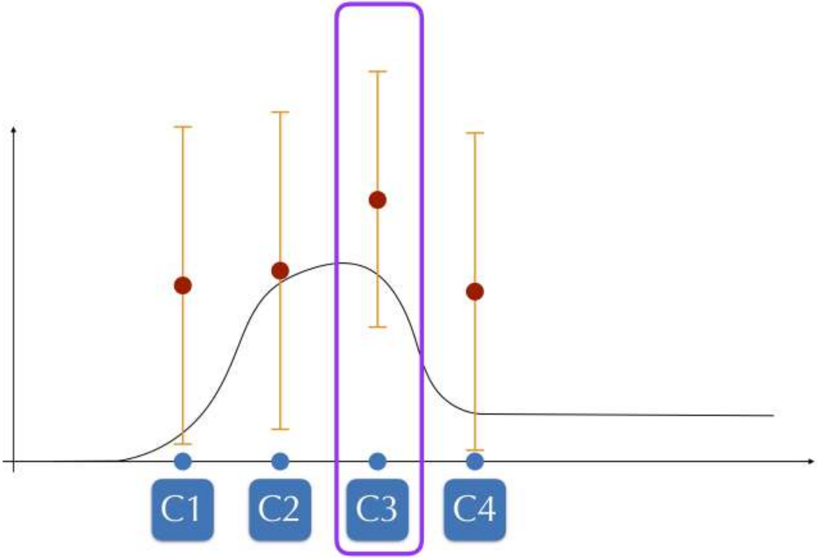
Failure of UCB1



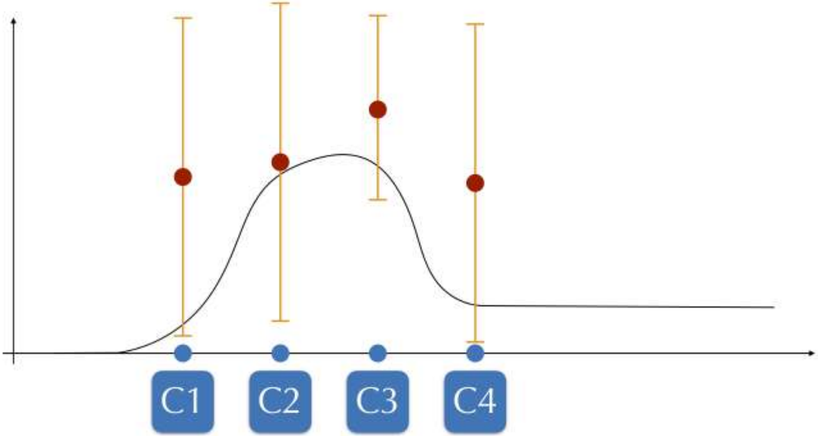
Failure of UCB1



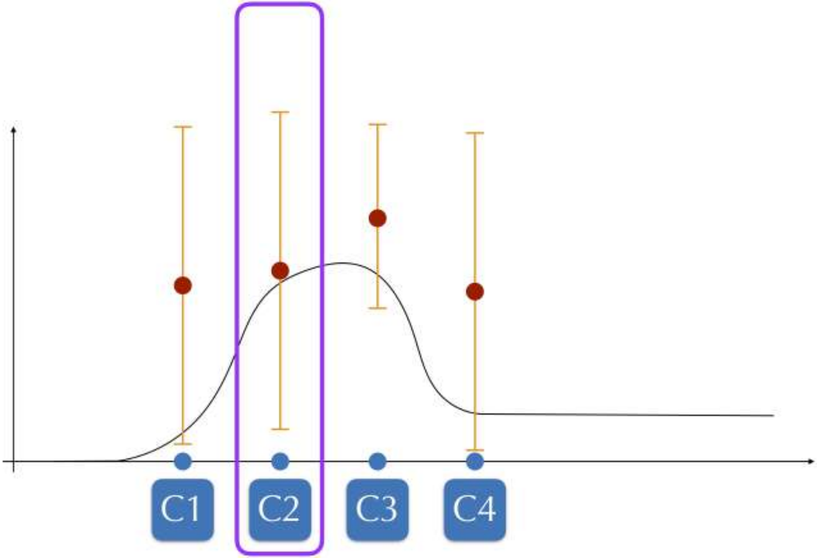
Failure of UCB1



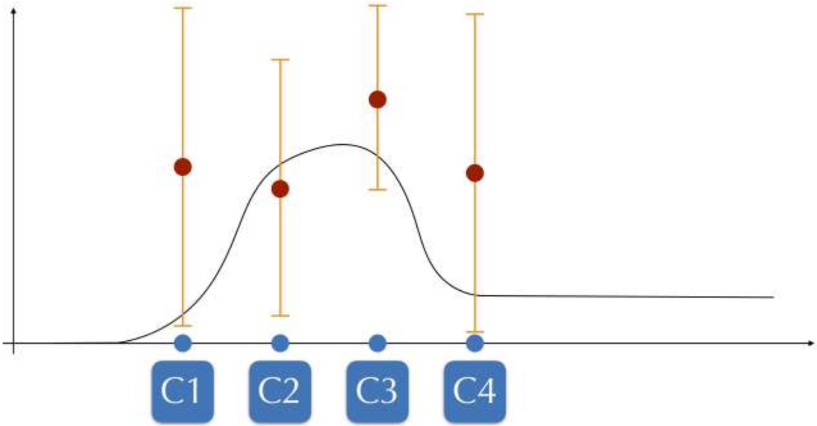
Failure of UCB1



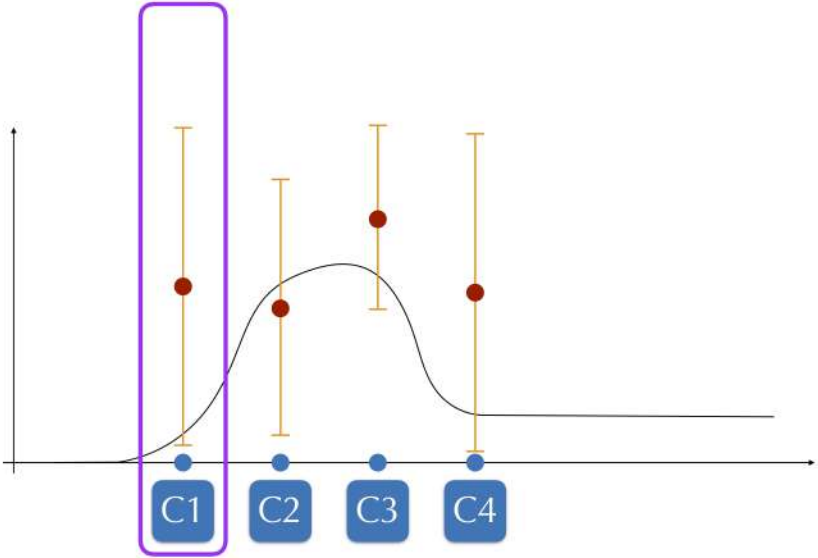
Failure of UCB1



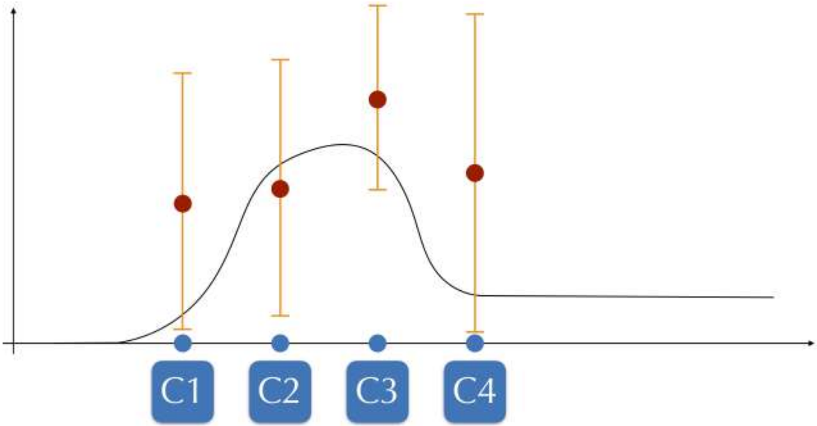
Failure of UCB1



Failure of UCB1

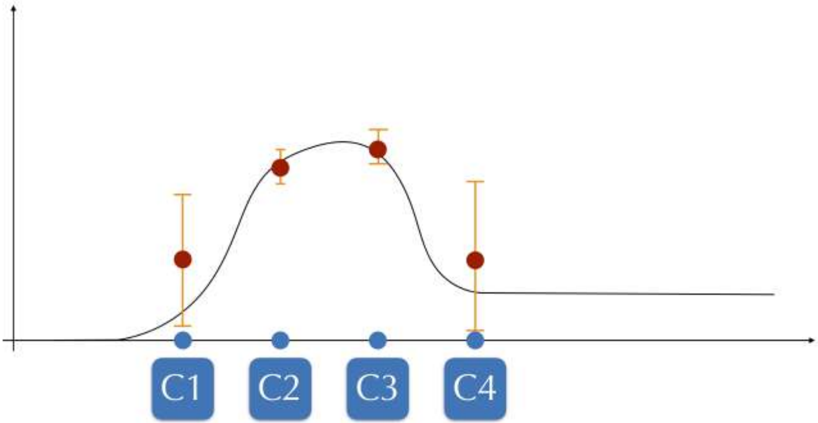


Failure of UCB1

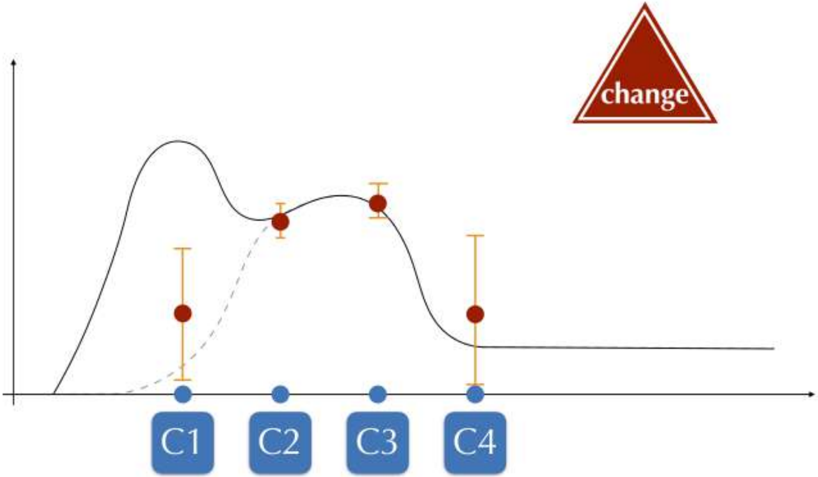


Failure of UCB1

(after a lot of samples)



Failure of UCB1



Non-stationary algorithms

Sliding window

Idea: consider only **recent samples**.

We use a **sliding window** of size W to forget old samples:

- We compute the estimate of the algorithm (e.g., UCB1 or TS) using the last $W < T$ samples
- **Advantage:** this guarantees that the bounds do not monotonically reduce over time
- **Drawback:** we do not exploit all the samples

Sliding window

Idea: consider only **recent samples**.

We use a **sliding window** of size W to forget old samples:

- We compute the estimate of the algorithm (e.g., UCB1 or TS) using the last $W < T$ samples
- **Advantage:** this guarantees that the bounds do not monotonically reduce over time
- **Drawback:** we do not exploit all the samples $\rightarrow$ the regret is larger in (almost) stochastic environments

UCB1 with sliding window

For each arm $a \in A$, we build a confidence interval around its empirical mean **using only the last W samples**.

We define:

- $N_{t,W}(a)$ as the number of time we pick arm a in the last W rounds at time t:

$$N_{t,W}(a) := \sum_{t'=\max\{t-W,0\}}^{t} \mathbb{I}[a_{t'} = a].$$

- $\mu_{t,W}(a)$ as the empirical mean of arm a during the last W rounds:

$$\mu_{t,W}(a) = \frac{1}{N_{t-1,W}(a)} \sum_{t'=\max\{t-W,0\}}^{t-1} r_{t'}(a)\mathbb{I}[a_{t'} = a]$$

UCB1 with sliding window

For each arm $a \in A$, we build a confidence interval around its empirical mean **using only the last W samples**.

At each time $t \in [T]$, we define an UCB of the arm average reward $\mu_t(a)$:

$$UCB_t(a) = \underbrace{\mu_{t,W}(a)}_{exploitation\ \ term} + \underbrace{\sqrt{\frac{2\log(T)}{N_{t-1,W}(a)}}}_{exploration\ \ term}$$

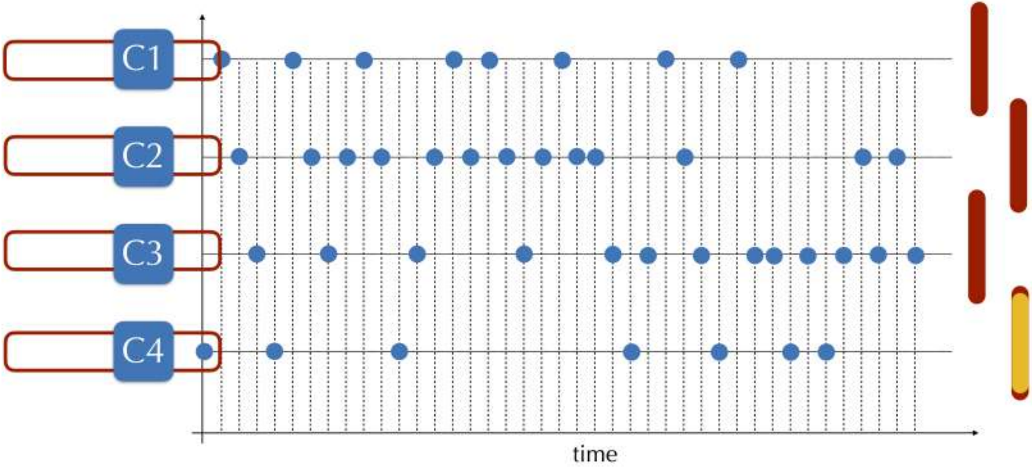
- The term $\mu_t(a)$ incentivizes to play arms with large empirical mean **in the last W rounds**
- The term $\sqrt{\frac{2\log(T)}{N_{t-1,W}(a)}}$ incentivizes to play arms with low $N_{t-1,W}(a) \rightarrow$ played a small number of times **in the last W rounds**

UCB1 with sliding window

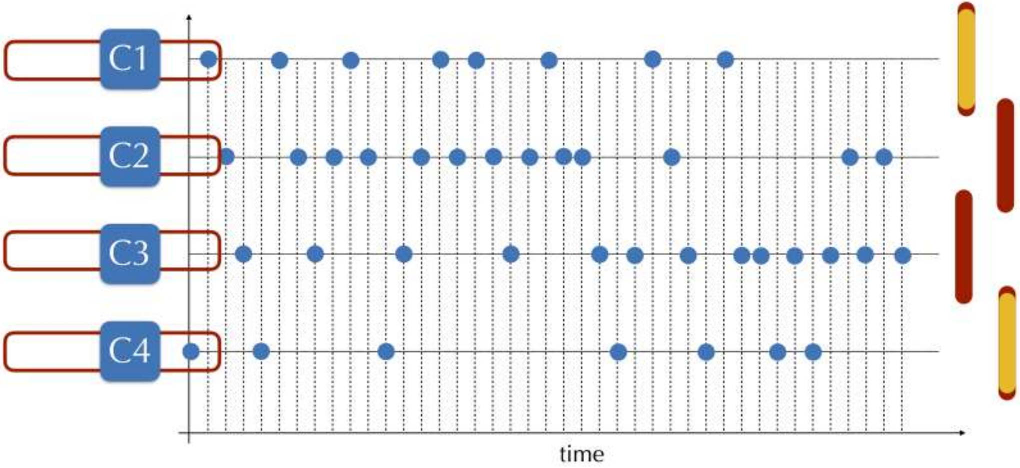
Algorithm: UCB1 with sliding window

set of arms A, number of rounds T, **sliding window** W;
for $t = 1, \ldots, T$ **do**
  for $a \in A$ **do**
    $\mu_{t,W}(a) \leftarrow \frac{1}{N_{t-1,W}(a)} \sum_{t'=\max\{t-W,0\}}^{t-1} r_{t'}(a)\mathbb{I}[a_{t'} = a]$;
    $UCB_t(a) \leftarrow \mu_t(a) + \sqrt{\frac{2\log(T)}{N_{t-1,W}(a)}}$;
  play arm $a_t \in \arg\max_a UCB_t(a)$;

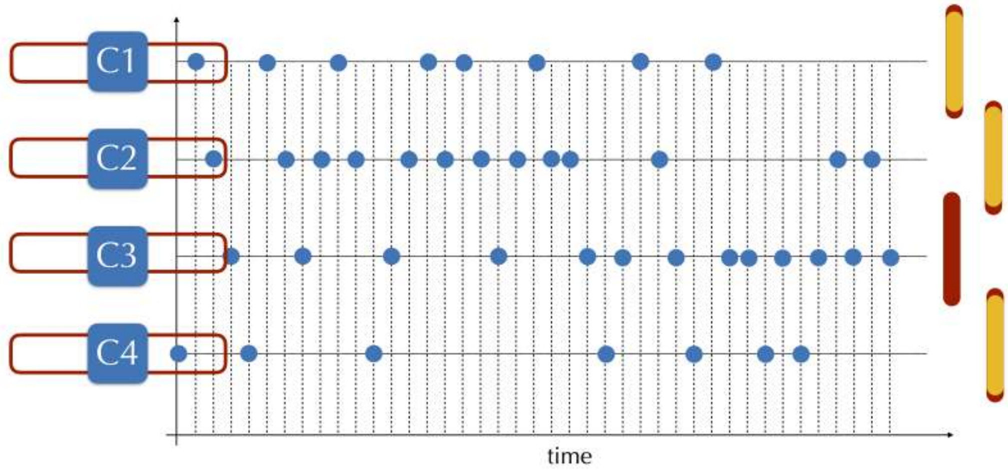
Example of UCB1 with sliding window



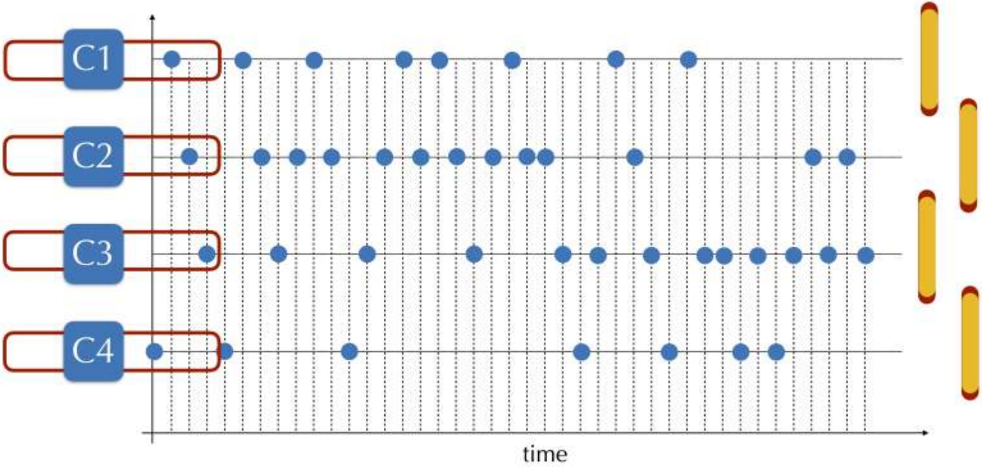
Example of UCB1 with sliding window



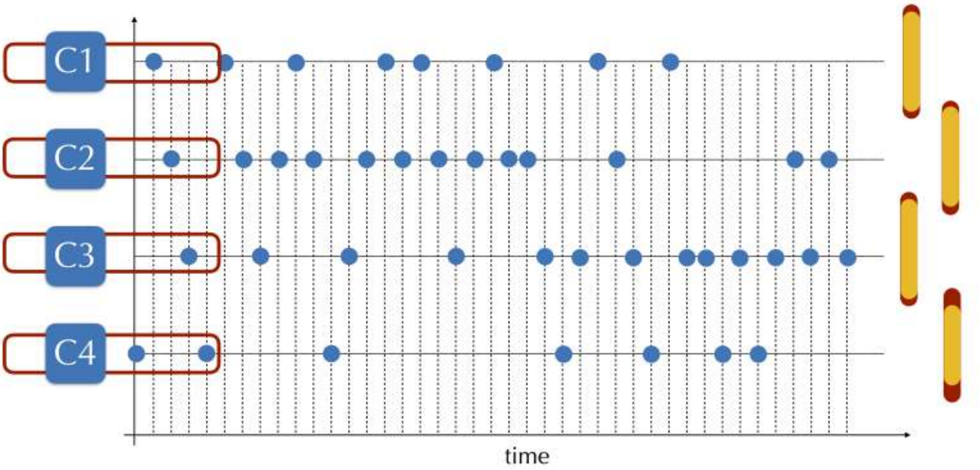
Example of UCB1 with sliding window



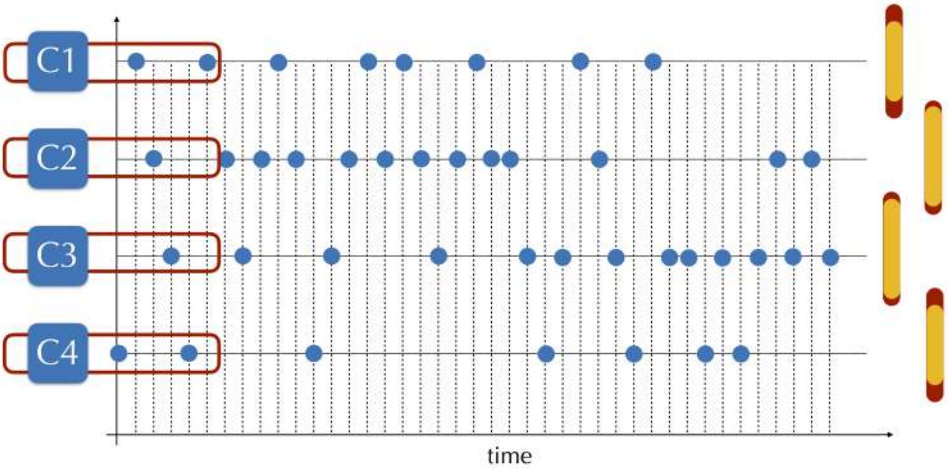
Example of UCB1 with sliding window



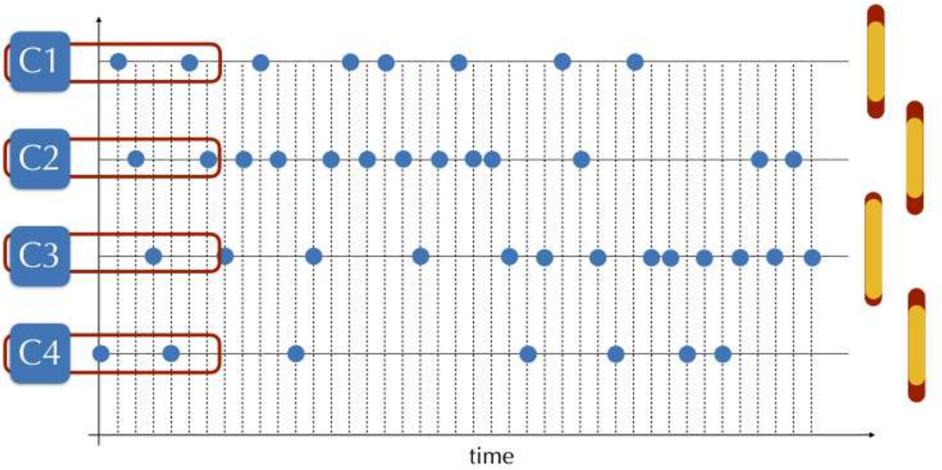
Example of UCB1 with sliding window



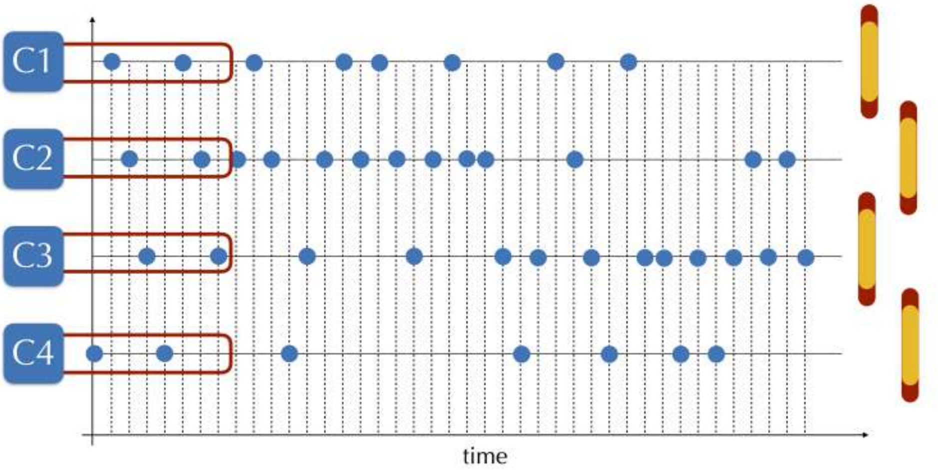
Example of UCB1 with sliding window



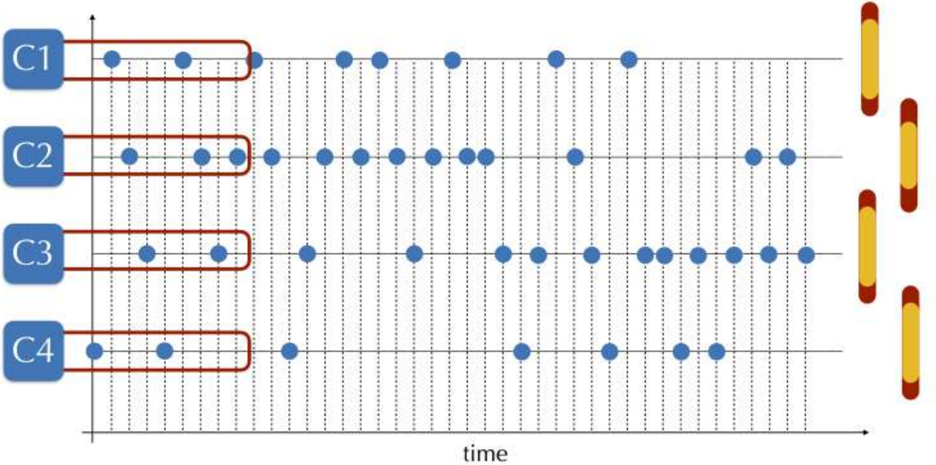
Example of UCB1 with sliding window



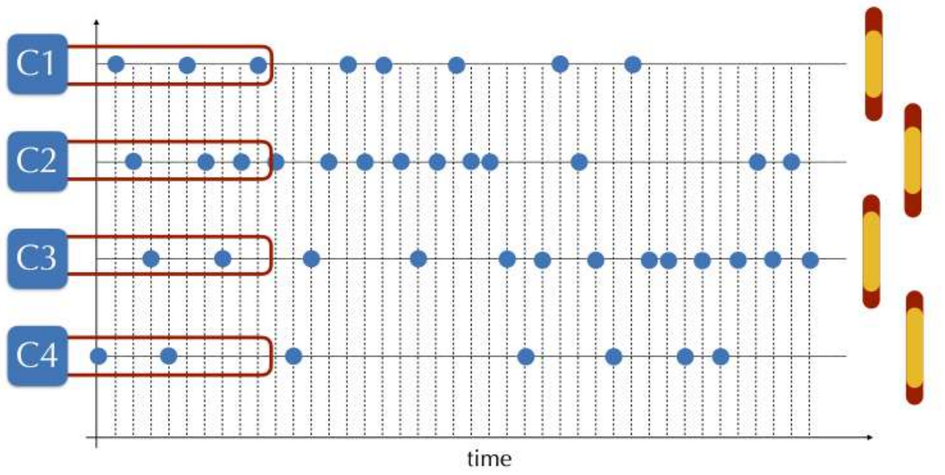
Example of UCB1 with sliding window



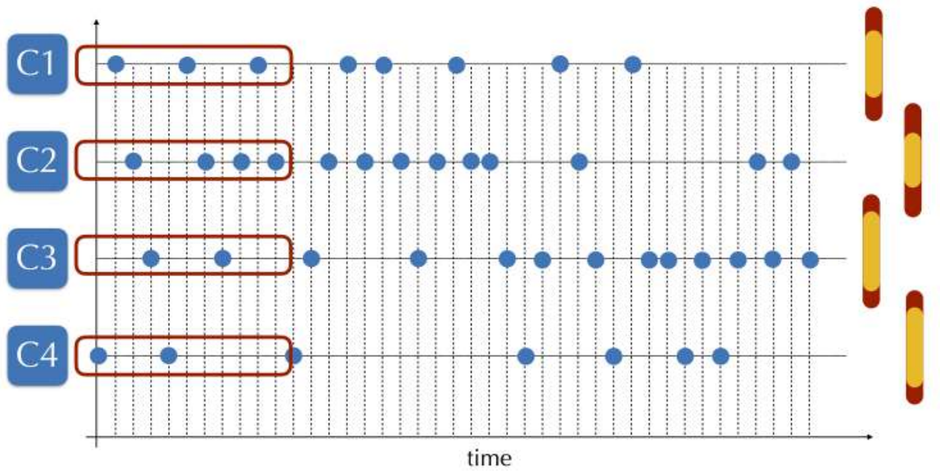
Example of UCB1 with sliding window



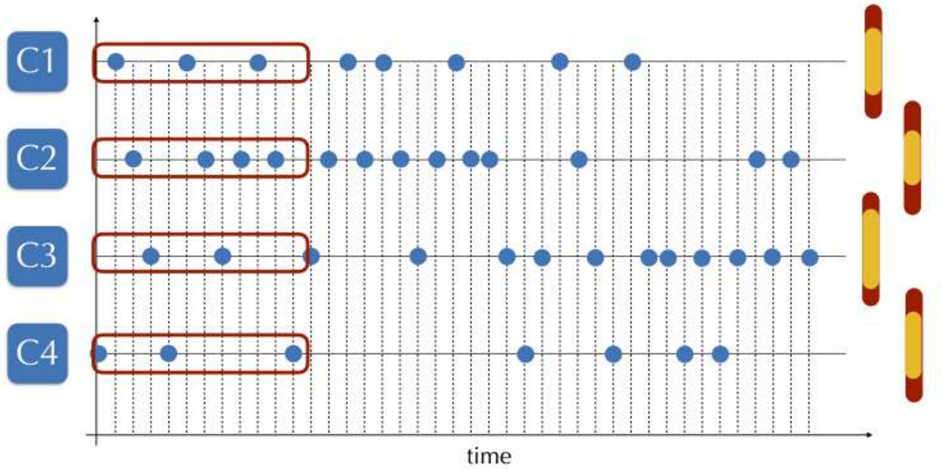
Example of UCB1 with sliding window



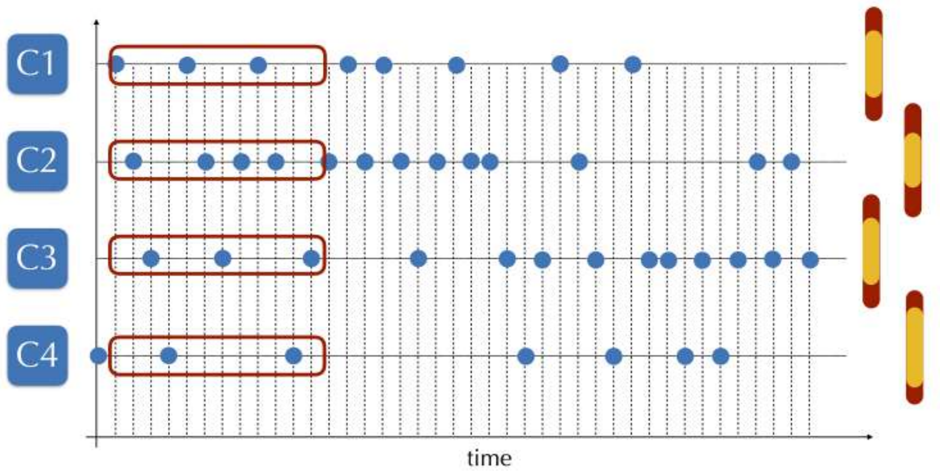
Example of UCB1 with sliding window



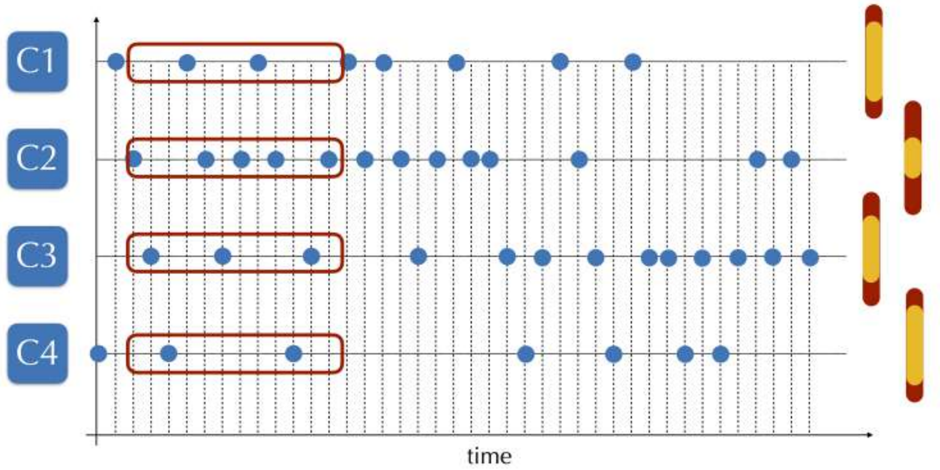
Example of UCB1 with sliding window



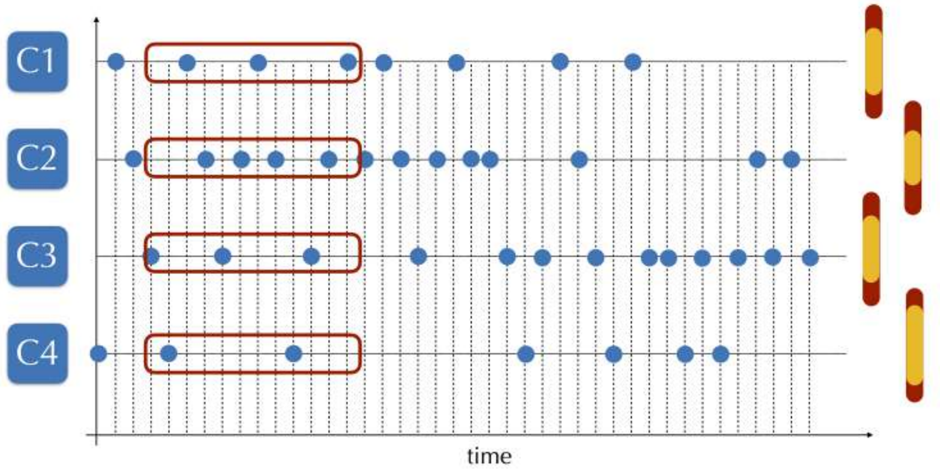
Example of UCB1 with sliding window



Example of UCB1 with sliding window



Example of UCB1 with sliding window



Thompson sampling with sliding window

We can also combine a sliding windows (of size W) with **Thompson sampling**:

- We focus on Bernulli reward distributions (i.e., with support $\{0, 1\}$)
- For every arm, have a prior distribution on its expected value
- For every arm, draw a sample according to the corresponding prior distribution
- Choose the arm with the best sampled reward
- Update the prior distribution of the chosen arm according the observed realization **considering only the last W samples**

Update the distribution

- We start with prior $Beta(1,1) \rightarrow$ uniform distibution over $[0,1]$
- When we observe a new sample we increase α if we observe $r_t(a) = 1$ or β if we observe $r_t(a) = 0$
- **We remove the samples that are older than W rounds**

$$(\alpha_a, \beta_a) \leftarrow (\alpha_a, \beta_a) + (r_t(a), 1 - r_t(a))\mathbb{I}[a_t = a] - (r_{t-W}(a), 1 - r_{t-W}(a))\mathbb{I}[a_{t-W} = a],$$

where we consider the last term only if $W > t$.

Choose which arm to play

- For each arm a, sample $\theta_a \sim Beta(\alpha_a, \beta_a)$
- Play the arm with the largest sampled mean $a_t \in \arg\max_{a \in A} \theta_a$

Thompson sampling with sliding window

Algorithm: Thompson sampling with sliding window

set of arms A, number of rounds T;
for $a \in A$ **do**
  $\alpha_a = \beta_a = 1$;
for $t = 1, \ldots, T$ **do**
  for $a \in A$ **do**
    $\theta_a \sim Beta(\alpha_a, \beta_a)$;
  play arm $a_t \in \arg\max \theta_a$;
  for $a \in A$ **do**
    if $t \leq W$;
    then
      update $(\alpha_a, \beta_a) \leftarrow (\alpha_a, \beta_a) + (r_t(a), 1 - r_t(a))\mathbb{I}[a_t = a]$;
    else
      update $(\alpha_a, \beta_a) \leftarrow$
      $(\alpha_a, \beta_a) + (r_t(a), 1 - r_t(a))\ \mathbb{I}[a_t = a] - (r_{t-W}(a), 1 - r_{t-W}(a))\mathbb{I}[a_{t-W} = a]$;

Concluding remarks on sliding windows

Some final remarks on sliding windows methods:

- The best window size W depends on the specific **unknown** instance
- A good choice in practice is to set $W \simeq \sqrt{T}$
- Sliding window forces every arm to be chosen repeatedly during time
- Every arm is re-evaluated during time
- Non-stationary environments require permanent exploration
- The regret is much higher than stochastic settings

Change detection

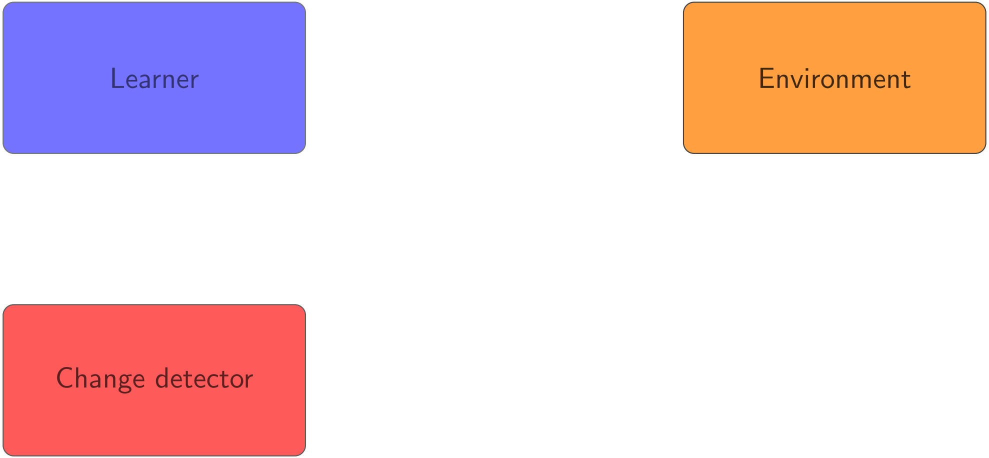
Change detection

- Sliding window deals with non-stationary environments **passively**
- Sliding window does not identify when the environment changes

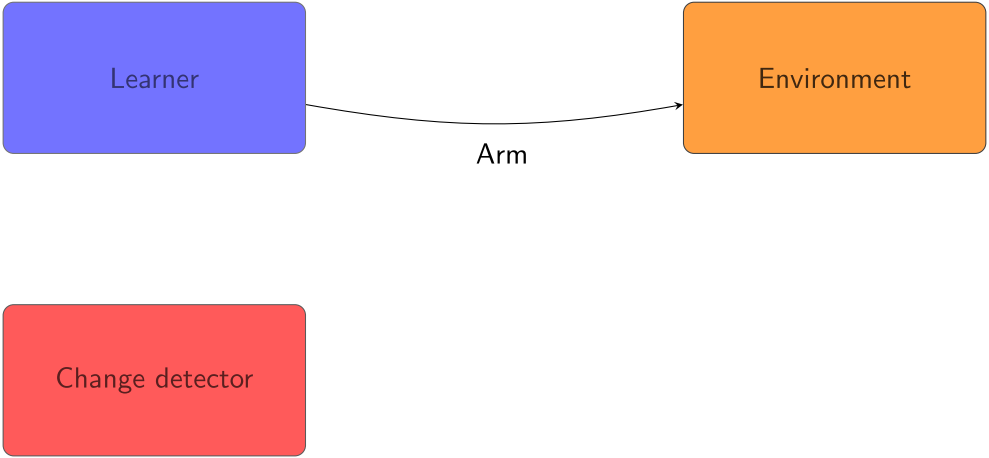
We can implement more **active** approaches to deal with non-stationary:

- Design an addition component that **detects** changes in the environment
- Reset the learning algorithm when a change is detected
- The learning algorithm observes an almost-stationary environment

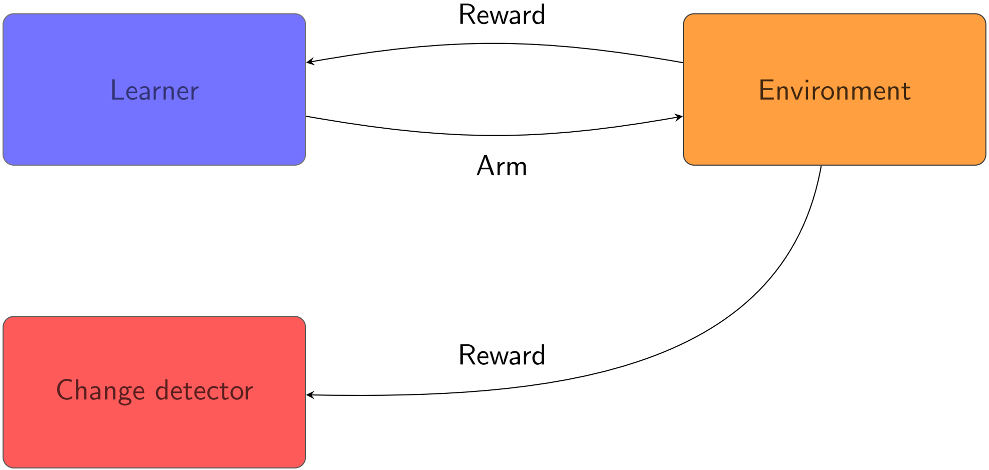
Change detection



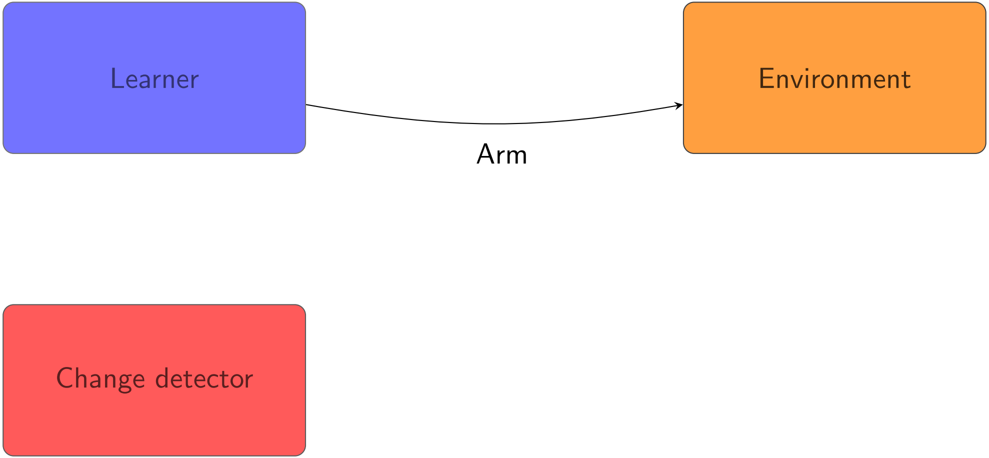
Change detection



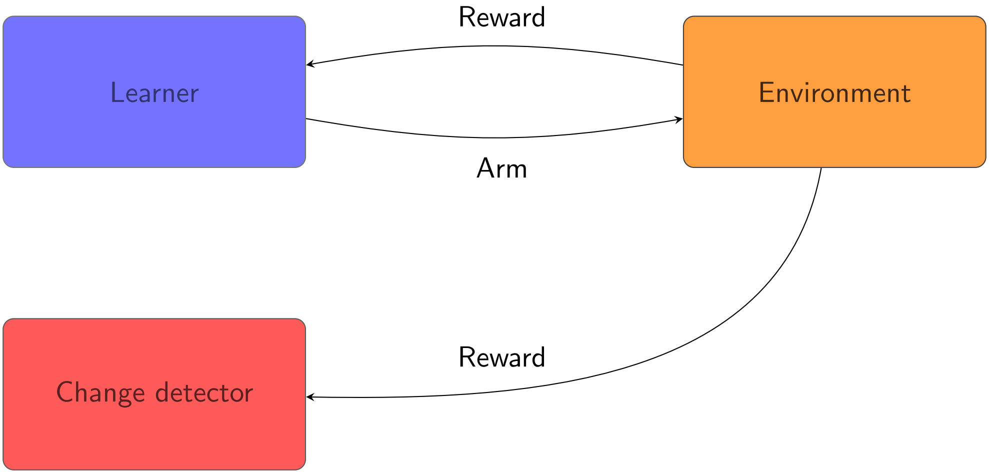
Change detection



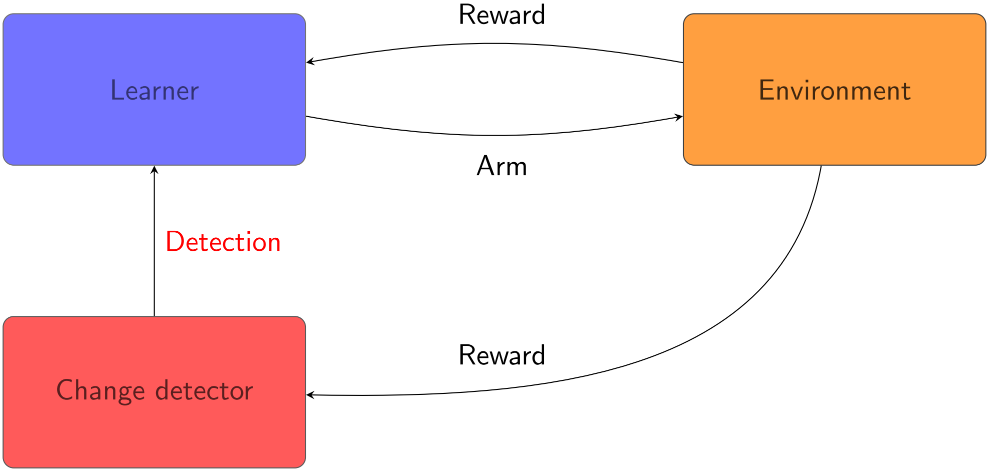
Change detection



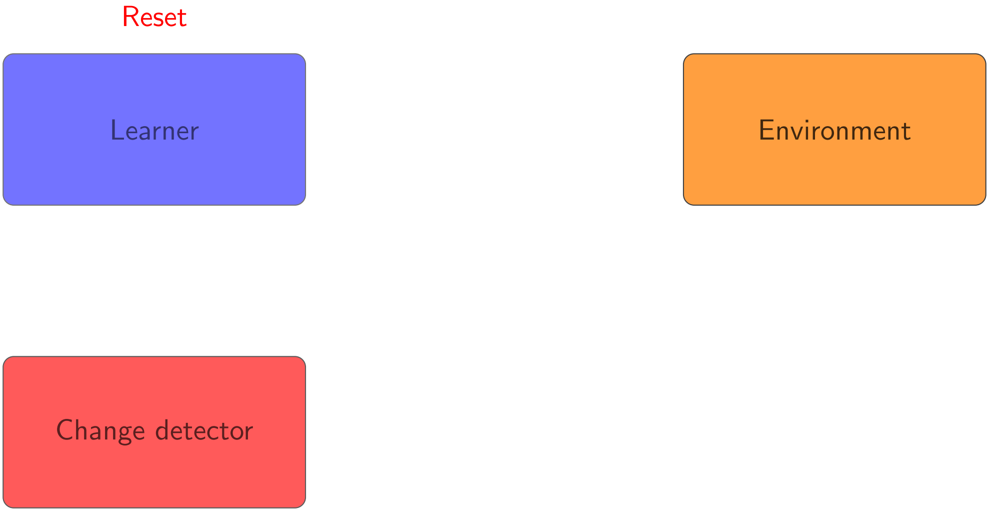
Change detection



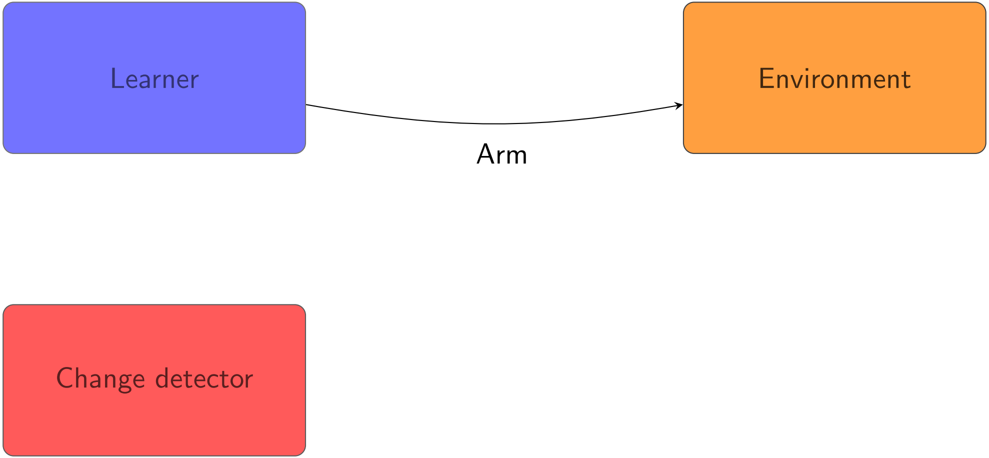
Change detection



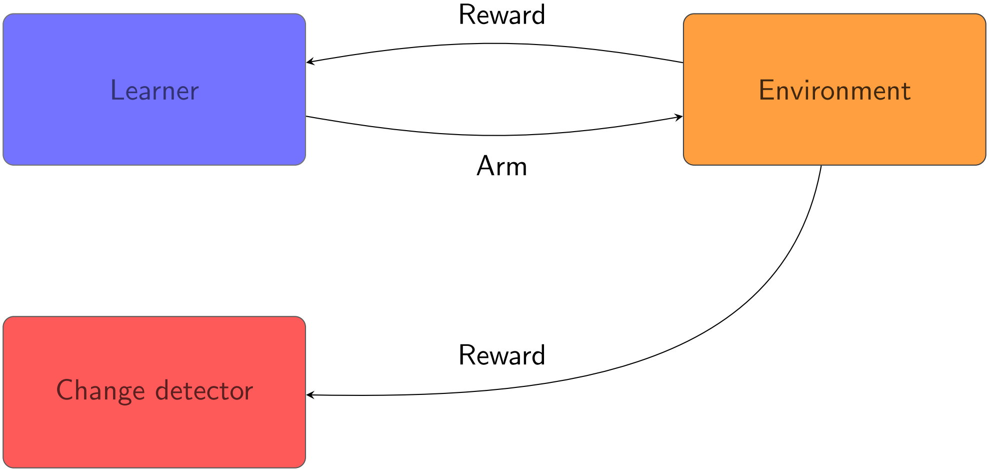
Change detection



Change detection



Change detection



UCB1 with change detection

For each arm $a \in A$, we build a confidence interval around its empirical mean **using only the samples after the last change detection**.

We define:

- τ_a be the last change detection for arm a ($\tau_a = 1$ if no change detect)
- $N_t(a)$ as the number of times we pick arm a after the last change detection at time t:

$$N_t(a) := \sum_{t'=\tau_a}^{t} \mathbb{I}[a_{t'} = a].$$

- $\mu_t(a)$ as the empirical mean of arm a after the last change detection:

$$\mu_t(a) = \frac{1}{N_{t-1}(a)} \sum_{t'=\tau_a}^{t-1} r_{t'}(a)\mathbb{I}[a_{t'} = a]$$

UCB1 with change detection

For each arm $a \in A$, we build a confidence interval around its empirical mean **using only the samples after the last change detection**.

At each time $t \in [T]$, we define an UCB of the arm average reward $\mu_t(a)$:

$$UCB_t(a) = \underbrace{\mu_t(a)}_{exploitation\ term} + \underbrace{\sqrt{\frac{2\log(T)}{N_{t-1}(a)}}}_{exploration\ term}$$

- The term $\mu_t(a)$ incentivizes to play arms with large empirical mean **after the last change detection**
- The term $\sqrt{\frac{2\log(T)}{N_{t-1}(a)}}$ incentivizes to play arms with low $N_{t-1}(a) \rightarrow$ played a small number of times **after the last change detection**

UCB1 with change detection

For each arm $a \in A$, we build a confidence interval around its empirical mean **using only the samples after the last change detection**.

At each time $t \in [T]$, we define an UCB of the arm average reward $\mu_t(a)$:

$$UCB_t(a) = \underbrace{\mu_t(a)}_{exploitation\ \ term} + \underbrace{\sqrt{\frac{2\log(T)}{N_{t-1}(a)}}}_{exploration\ \ term}$$

- The term $\mu_t(a)$ incentivizes to play arms with large empirical mean **after the last change detection**
- The term $\sqrt{\frac{2\log(T)}{N_{t-1}(a)}}$ incentivizes to play arms with low $N_{t-1}(a) \rightarrow$ played a small number of times **after the last change detection**

⚠ With small probability we explore uniformly. **This guarantees to explore in order to detect if the rewards of any arm is changing.**

UCB1 with change detection

Algorithm: $\mathrm{UCB1}$ with change detection

set of arms A, number of rounds T, **probability α, change detector CD**;
$\tau_a \leftarrow 1$ for each $a \in A$;
for $t = 1, \ldots, T$ **do**
    for $a \in A$ **do**
        $\mu_t(a) \leftarrow \frac{1}{N_{t-1}(a)} \sum_{t'=\tau}^{t-1} r_{t'}(a)\mathbb{I}[a_{t'} = a]$;
        $UCB_t(a) \leftarrow \mu_t(a) + \sqrt{\frac{2\log(T)}{N_{t-1}(a)}}$;
    $a_t \leftarrow \arg\max_{a \in A} UCB_t(a)$;
    $a_t \leftarrow$ random arm with probability α;
    if $CD_{a_t}(r_{\tau_{a_t}}, \ldots, r_t) = 1$ **then**
        $\tau_{a_t} \leftarrow t$ (**this resets the change detector and UCB1 for arm a_t**)

TS with change detection

Similarly, we can extend Thompson sampling with change detection. As for UCB with change detection, we restart the estimation of arm a whenever a change for this arm is detected.

CUSUM: a simple change detector

Change detection with CUSUM

A simple change detector is **CUSUM**.

For every arm a, the change detector works in two phases:

Estimation phase

Used to build an estimation of the reward of the arm $\mu(a)$.

Detection phase

Used to detect if there is a significant change from the estimated distribution.

Estimation phase

Estimation phase

- Use the first M samples of the reward of arm a to compute the the empirical mean $\bar{\mu}(a)$ of the arm
- M is a parameter to set

Detection phase

Detection phase

- For any additional sample of the reward of arm a check if there is a significant change from the empirical mean
- Compute the positive deviation from the estimated mean $\bar{\mu}(a)$

$$s_a^+ = r_t(a) - \bar{\mu}(a) - \epsilon$$

- Compute the negative deviation from the estimated mean $\bar{\mu}(a)$

$$s_a^- = -(r_t(a) - \bar{\mu}(a)) - \epsilon$$

- ϵ is a parameter to set
- ϵ determines how large should be a change to be detected

Detection phase

Detection phase

- Update the cumulative deviations from the estimated mean

$$g_a^+ = \max\{0, g_a^+ + s_a^+\}$$

$$g_a^- = \max\{0, g_a^- + s_a^-\}$$

- g_a^+ and g_a^- are initialized to 0 when the change detection is initialized
- **We detect a change if $g_a^+ > h$ or $g_a^- > h$**
- h is a parameter to set