From Routine Bandits to Non Stationary Bandits

Mustapha AJEGHRIR | Asmae KHALD | Nouamane TAZI

CentraleSupélec – Département Informatique

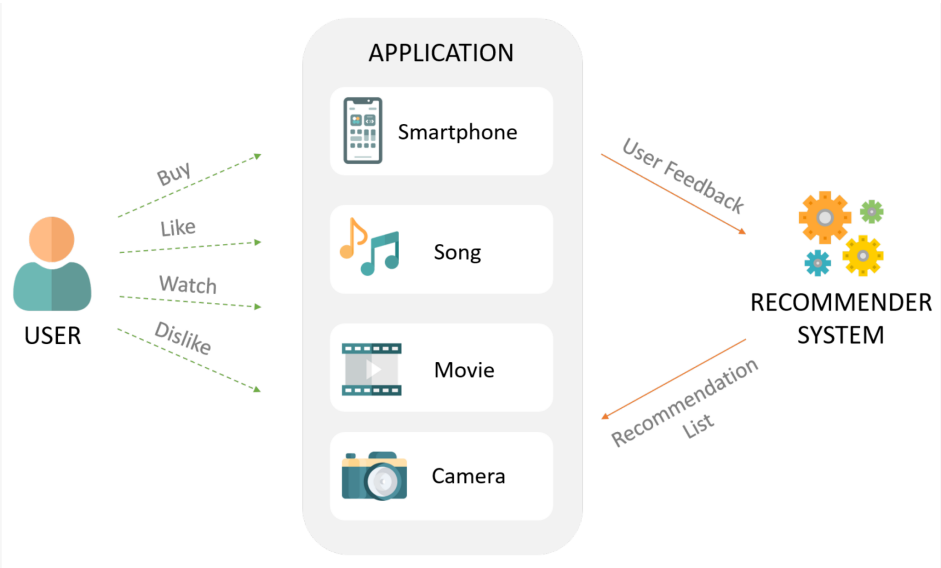
5 avril 2022

Problem : Recommender Systems

Recommender systems

When a recommender system is deployed on multiple users, one does not typically assume that the best recommendation is the same for all users. The naive strategy

Figure – Recommander system



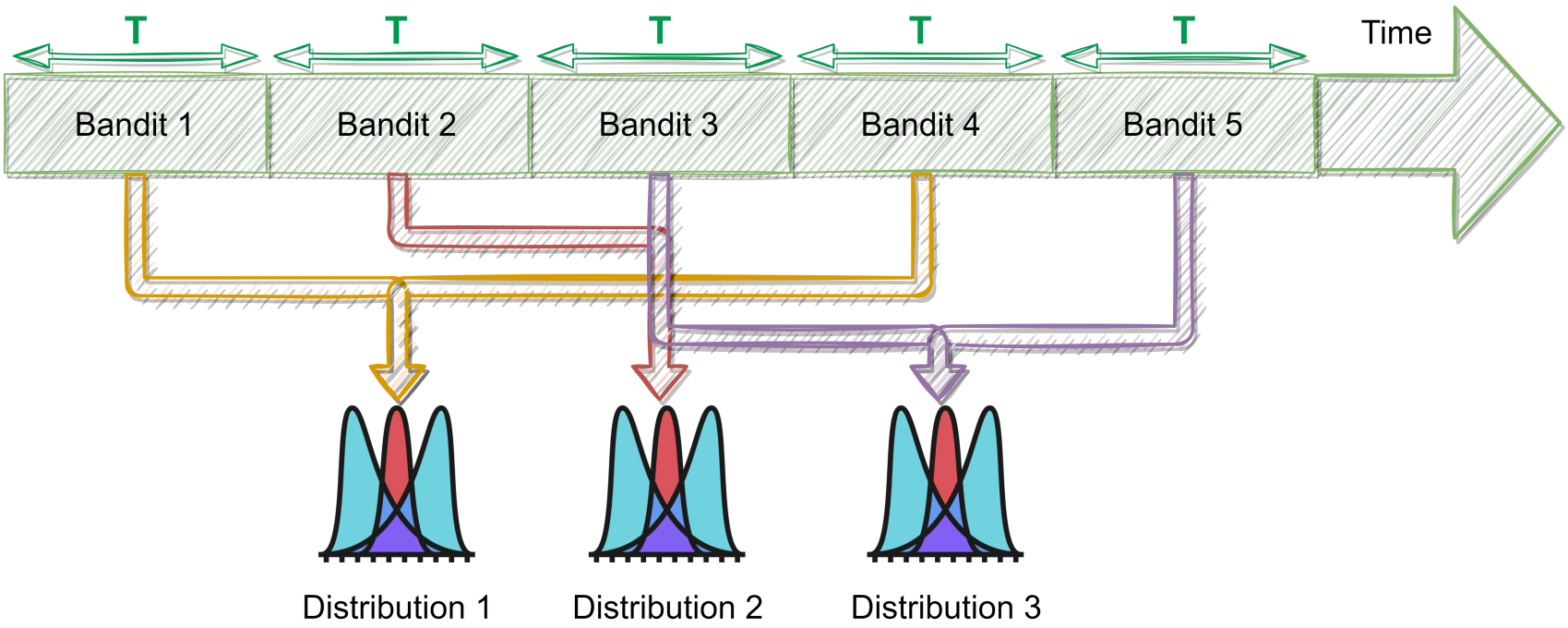
Routine Bandits

Routine Bandits : definition

The Horizon T is known to the learner and the possible distributions are finite.

The learner aims to reuse previously seen bandits.

Figure – Routine Bandits

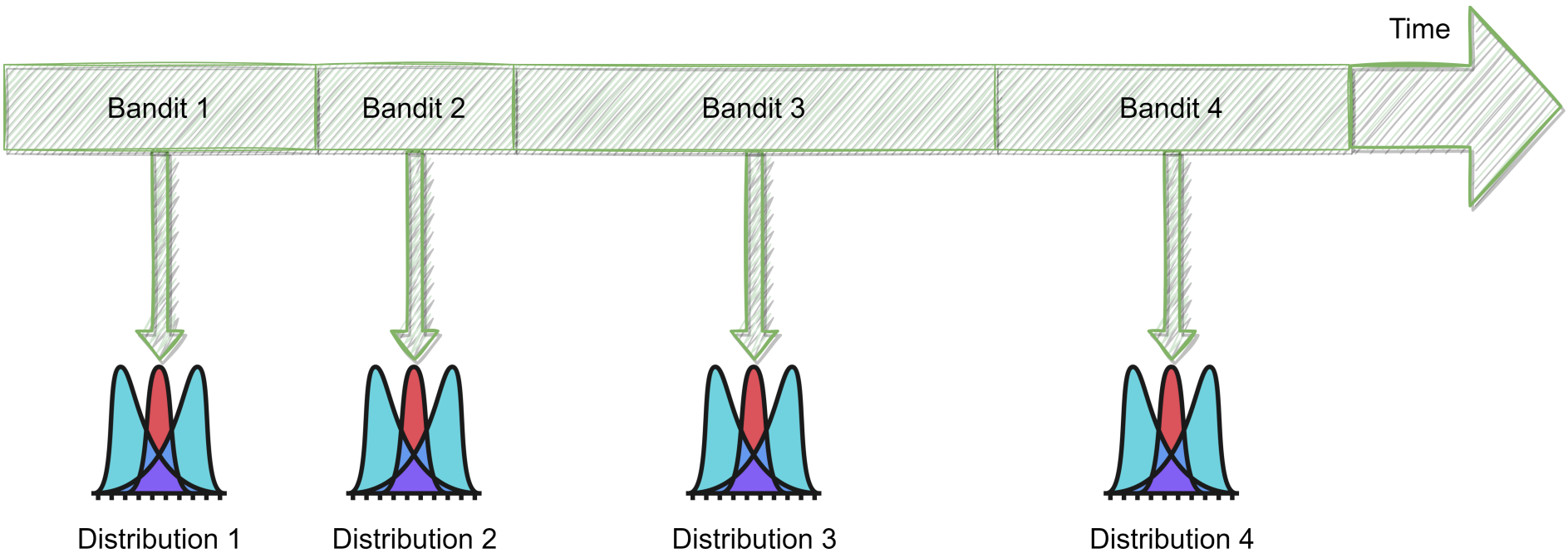


Non Stationary bandits

Non Stationary Bandits : definition

The length of each bandit is unknown, the learner aims to detect when a new bandit arrives.

Figure – Non Stationary Bandits



Upper Confidence Bound

Arm selection with UCB

$$U_a(t) = \hat{\mu}_a(t) + \sqrt{\frac{\log(1/\delta_t)}{2N_a(t)}},$$

with $\delta_t = t^{-2} \times (t+1)^{-1}$ and :

N_a : number of times a is drawn

$\hat{\mu}_a$: empirical mean for arm a

Arm selection with KL_UCB

$$U_a(t) = \sup \left\{ \mu \in \overline{I} : d\left(\widehat{\mu}_a(t), \mu\right) \leq \frac{\ln(t)}{N_a(t)} \right\}$$

d : is the kullback leibler divergence

$\overline{I}$: μ's interval, for Bernoulli it is equal to $[0, 1]$

Upper Confidence Bound Variants

R_UCB or R_KL_UCB

Refresh the learner every $T_R := \textit{period}$ steps.

- Every T_R steps, remove all the internal variables and reinitialize the learner.
- This learner is very sensitive to the parameter T_R.

SW_UCB or SW_KL_UCB

Uses a sliding window as a Buffer of size T_B.

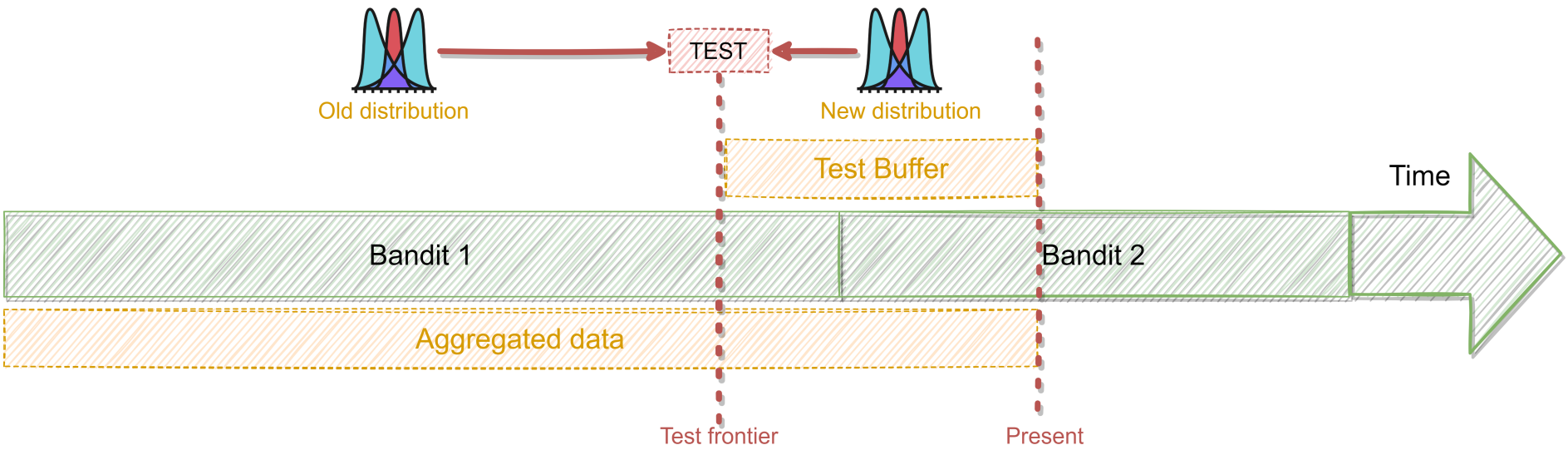
- Use FIFO data structure to save the last T_B draws.
- Resilient to the lake of knowledge of the real bandits' Lengths.

Non stationary Upper Confidence Bound

NS_UCB or KL_NS_UCB

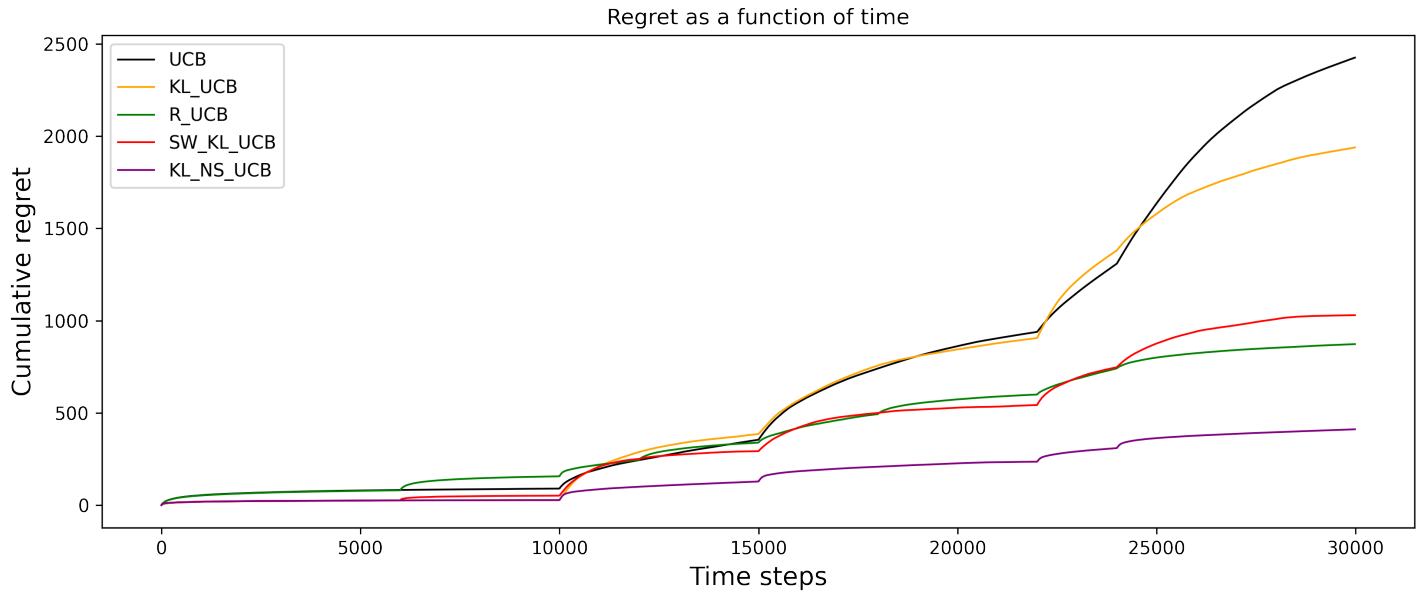
Uses a sliding window of size T_B as a buffer to continuously test if the new inferred distribution is compatible with the old one.

- Supports a long range of bandits' Lengths.
- Best Results.
- could be computationally expansive.



Comparing algorithms

Figure – Comparing main algorithms




```
timeHorizon = 30000
N_exp       = 100
nbArms      = 5
ts          = [10000, 5000, 7000, 2000]
```

Algorithm	Period	Buffer_size
UCB	-	-
KL_UCB	-	-
R_UCB	6000	-
SW_KL_UCB	-	6000
KL_NS_UCB	-	150

Conclusion

The KL_NS_UCB algorithm performs better than all the previously seen algorithms, but with larger hardware overhead. The best thing in this algorithm is we don't need to know any approximation of the bandits' lengths to yield a good performance. On the other hand, SW_KL_UCB is also a good algorithm for applications where efficiency is important.