Akram Erraqabi

ENSAE 3ème année
Stage de fin d'études
2014–2015

RAPPORT DE STAGE

Error and Regret Trade Off for a Multi Armed Bandit Problem

Inria Lille – Nord Europe Maître de stage: Michal Valko
Co-encadré par A. Lazaric, E. Brunskill
Lille, France

11/05/2015 – 13/11/2015

Lille, Septembre 2015

Contents

1 Trading off Error and Regret **3**
1.1 Setting 4
1.2 Some learning algorithms 6
1.2.1 An explore/exploit approach 6
1.2.2 Forcing algorithm & Regularization 7
1.2.3 Thompson Sampling 7
1.2.4 A UCB-like algorithm 8
1.3 Empirical Analysis of UCB-REGERR the case of 2 gaussian arms . 9
1.3.1 Curse of unbalanced allocation for UCB-REGERR 10
1.3.2 Regret Empirical Analysis 11

Part 1

Trading off Error and Regret

Crowdsourcing is a cheap and efficient method to provide solutions to simple tasks that are difficult for computers but possible for humans. A crowdsourcing platform gathers *requesters*, who need to have their tasks done, and *workers* who are ready to perform these tasks in exchange of payment. The interactive feature of crowdsourcing markets attracts researchers in order to design adaptive algorithms able to optimize various aspects of these markets, such as the pricing and the assignment of tasks. The later is a motivation of the present work. Indeed, the crowdsourcing platform values the skills and the preferences of the workers and the expectations and the goals of the requesters in order to assign the right task to the most "appropriate" worker. This work stands as an attempt of fitting a Multi-armed Bandit setting to a Crowdsourcing context.

Multi-armed Bandit problem (MAB) is the classical setting of an agent that aims at optimizing his choice (or strategy) among a set of decisions. These decisions are associated to the *arms* of the problem. Each arm is characterized by a distribution and pulling the arm means drawing a sample from that distribution. This sample is considered as an observed reward (or loss). Formally, a MAB problem of size K consists of K probability distributions $D_1, ..., D_K$ with expected values $\mu_1, ..., \mu_K$. The D_i and μ_i are not known at the start. These distributions are classically viewed as wins or losses (1 or 0) from various arms of a slot machine, but in continuous formulations can be any (bounded) user-defined function.

In the context of a *crowdsourcing* platform, we can think of the arms as different workers and the reward as the quality of the job assigned to the worker. Obviously, from a task category to another the arm distributions may change since a worker that performs bad for category A may perform well for category B. However, the platform must also keep the worker busy (and thus earning money) otherwise he leaves the market and this one shrinks. One can imagine that the platform shouldn't always assign a task to the worker with the best recorded performance. It may prefer to give it to a worker that is closer to the best one but that performs bad for other tasks. Thus, in order to keep the worker in the market, the platform will provide the requester with a close to optimal quality work.

The subtlety here is that it's neither about selecting the best arm for which

some algorithms were suggested in [2, 6] nor about a pure exploration to estimate of the arms means (see [5, 1]). The goal is to identify good arms (high means) while guaranteeing enough estimation precision for less interesting ones. Our research led us to a comparable approach used for *curriculum design* or more precisely for user learning in a MOOC[1] context where teaching techniques have to be evaluated without impeding the user's learning experience (see [7]). In this part, we present our attempt to design a solution inspired by the work in [7]. Then we describe some approaches (algorithms) that have been derived for this task. We focus our empirical analysis on the UCB-REGERR algorithm that we suggest through this work, in order to evaluate its behavior in different contexts. These results turn out to be satisfying enough as they prove the empirical consistency of the algorithm and thus invite us to derive theoretical analysis to confirm the algorithm performance.

1.1 Setting

A mixed objective function

Brunskill et al. considered, in [7], an objective function that mixes the (cumulative) reward and the estimation error of the arms means as in the following:

Let's assume that we dispose of an horizon of T steps and K arms, μ and σ are vectors of arms' means and standard deviation. We denote I_t the arm pulled at time t, $r_{I_t,t}$ the reward generated from this pull, Δ_i^T the size of the 95% confidence interval of arm i at time T. For every weight $w \in [0,1]$, we define the objective function as

$$f_w(\lambda; \mu, \sigma) = w \sum_{t=1}^{T} r_{I_t,t} - (1-w) \sum_{i=1}^{K} \Delta_i^T \tag{1.1}$$

That is, we want to maximize the total reward received, but minimize the sizes of the 95% confidence intervals size with some weight w between both goals. w helps the experimenter to balance arbitrary the importance of the estimation precision with respect to reward.

Let $L_{i,t} = \mathbb{E}\left[(\hat{\mu}_{i,t} - \mu_i)^2\right]$ the expected squared estimation error of the arm i up to time step t. $\hat{\mu}_{i,t}$ is the empirical mean of arm i up to time t. We first decided to change the precision measure to the expected square loss and rescale the sums in order to deal with reduced quantities. By taking the expectation of the reward term, we get

$$f_w(\lambda; \mu, \sigma) = w\, \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} r_{I_t,t}\right] - (1-w)\frac{T}{K}\sum_{i=1}^{K} L_{i,T} \tag{1.2}$$

Moreover, we denote $T_{i,t}$ the number of pulls of arm i up to time t and $\lambda_i = \frac{T_{i,T}}{T}$. If we suppose $T_{i,T}$ is known for every i, we can write $L_{i,T} = \frac{\sigma_i^2}{T_{i,T}}$ which allows writing

[1] Massive Open Online Course

the optimization problem as

$$\begin{aligned} \max_{\lambda} \quad & w \sum_{i=1}^{K} \lambda_i \mu_i - (1-w) \frac{1}{K} \sum_{i=1}^{K} \frac{\sigma_i^2}{\lambda_i} \quad = f_w(\lambda; \mu, \sigma) \\ \text{s.t} \quad & \sum_{i=1}^{K} \lambda_i = 1, \;\; \lambda_i \geq 0 \end{aligned} \tag{1.3}$$

Let $\boldsymbol{\lambda} = (\lambda_1, \cdots, \lambda_K)$. The optimal solution corresponds to an allocation of the pulls over the arms. Note that since $\lambda_i = \frac{T_{i,T}}{T}$ we have that $\boldsymbol{\lambda} \in \mathbb{Q}^K$. The previous problem formulation could be then considered as the relaxed version of the actual optimization problem. In the rest of the chapter, we will focus on that relaxed version.

Solutions of extreme cases

Taking the constraint $\boldsymbol{\lambda}^\top \mathbb{1} = 1$ into consideration, the first order condition can be written for an associated Lagrangian multiplier η as follows

$$\forall i, \;\; (w\mu_i - \eta) + \frac{1-w}{K} \frac{\sigma_i^2}{\lambda_i^2} = 0 \tag{1.4}$$

- Case 1: $w = 0$

  $$\forall i, \quad \frac{\sigma_i^2}{\lambda_i^2} = cte$$

  so

  $$\forall i \;\; \lambda_i = \frac{\sigma_i}{\sum_j \sigma_j}$$

  .

- Case 2: $w = 1$
  The problem is

  $$\max_{\lambda} \quad \sum_{i=1}^{K} \lambda_i \mu_i \quad \text{s.t} \quad \sum_{i=1}^{K} \lambda_i = 1, \;\; \lambda_i \geq 0 \tag{1.5}$$

  $$\lambda_i = \mathbf{1}_{i = \text{argmax}_j \mu_j}$$

When $0 < w < 1$, no explicit formula can be provided to the allocation $\boldsymbol{\lambda}$. Figure 1.1 gives the shape of that objective function with respect to w and λ in the case of two gaussian arms $K = 2$, where $\lambda = \lambda_1 = 1 - \lambda_2$. The setting is the following:

$$\mu = (1, 2) \quad \sigma = (1, 1)$$

For every w, the optimal allocation is located in red. Note that the extreme cases solutions are verifying the previous formula since when $w \to 0$ we see that $\lambda_1 = 0$ since $\mu_2 > \mu_1$ and for $w = 1$ we have $\lambda_1 = \frac{1}{2}$ since $\sigma_1 = \sigma_2$.

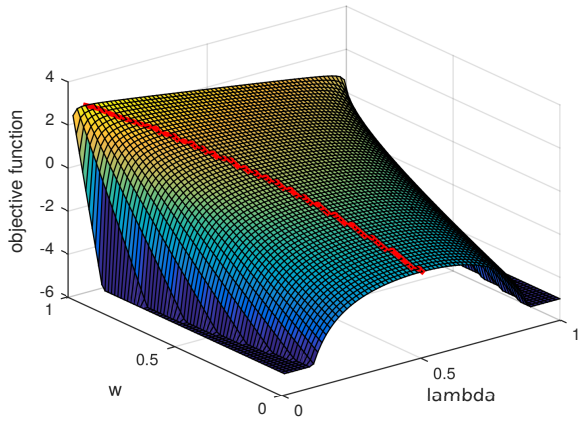


Figure 1.1: The shape of the objective function

1.2 Some learning algorithms

In this section, we present some learning algorithms that could be used for the present problem.

1.2.1 An explore/exploit approach

Let $\tau \in [0, T]$ be a time parameter ($\tau = \tau(T)$). The exploration and exploitation phases are separate. Up to time step τ, we pull an *underpulled* arm if it exists. At time step $t \leq \tau$, we say that an arm i is underpulled with respect to the exploration horizon τ, if:

$$T_{i,t} \leq \frac{c\sqrt{\tau}}{K}$$

where c is a parameter of the algorithm. (τ and c are parameters that may need some tuning.)

Algorithm 1 Explore/Exploit for Regret-Error Trade off

Parameters: τ, c, w, n, K.

Exploration Phase:
For the first τ steps: if there is still an underpulled arm, pull that one. If not, solve $\boldsymbol{\lambda_t} = \arg\max_\lambda \ f(\lambda, \hat{\mu}_t, \widehat{\sigma_t^2})$ and draw $I_t \sim \boldsymbol{\lambda_t}$
Exploitation Phase:
$\boldsymbol{\lambda_\tau} = \arg\max_\lambda f_w(\lambda, \hat{\mu}_\tau, \widehat{\sigma_\tau^2})$
for $t = \tau(T) + 1 \cdots T$ **do**
  Draw $I_t \sim \boldsymbol{\lambda_\tau}$ and pull arm I_t
  Update $\hat{\mu}_t$ and $\widehat{\sigma_t^2}$
end for

1.2.2 Forcing algorithm & Regularization

The idea of a forcing algorithm consist in forcing the algorithm to explore during the first steps by constraining the optimization problem to more exploratory solution. Then progressively relax the constraint and follow the natural path of the algorithm till the horizon T. This method helps avoiding to fall in a side solution from where the algorithm may not be able to escape.

The algorithm consists then in solving the optimization problem by plugging the empirical estimates of the parameters i.e the forcing algorithm considers at each time step t the following optimization problem:

$$\begin{aligned} \max_{\lambda} \quad & f_w(\lambda; \hat{\mu}_t, \widehat{\sigma_t^2}) = \; w \sum_{i=1}^{K} \lambda_i \hat{\mu}_{i,t} - (1-w) \frac{1}{K} \sum_{i=1}^{K} \frac{\widehat{\sigma_{i,t}^2}}{\lambda_i} \\ \text{s.t} \quad & \sum_{i=1}^{K} \lambda_i = 1 \; , \; \lambda_i \in [c(t), d(t)] \end{aligned} \tag{1.6}$$

where $c(t) \to 0$ and $d(t) \to 1$. At each time step, we draw the arm to be pulled according to the strategy λ solution of 1.6 that defines a distribution over the arms. The estimates are updated as new rewards are observed.

How should $c(t)$ and $d(t)$ converge ?

- Why shouldn't they converge exponentially fast? If the bounds $c(t)$ and $d(t)$ converge exponentially fast the forcing wouldn't be efficient when it could make the difference. The problem converges quickly to an unconstrained one.
- Why shouldn't they converge logarithmically? Here the convergence is very slow and will result in a higher regret.

The polynomial convergence $\frac{1}{\sqrt{t}}$ of the means and variances estimates may suggest the natural choice of $c(t) = 1 - d(t) = \frac{1}{\sqrt{t}}$.

1.2.3 Thompson Sampling

Here the estimation of the arms parameters μ and σ^2 is performed through a Thompson Sampling algorithm that consists in sampling these parameters from their posteriors. The latter are continuously updated while new observations are collected. Thompson sampling main effect is to push the posteriors to get more and more peaky around the true parameters value.

If we assume that (i) the rewards from a single arm, say arm i, are normally distributed $x^i \sim \mathcal{N}(\mu, \sigma^2)$, (ii) the mean $\mu \sim \mathcal{N}(\mu_0, n_0\sigma^2)$ and (iii) the variance $\sigma^2 \sim \text{Ga}(\alpha, \gamma)$, then the posterior distributions for arm i can be updated, after each

new sample x^i observed at the arm n^{th} pull, according to the following:

$$\mu \mid \sigma^2, x^i \sim \mathcal{N}\left(\frac{n}{n+n_0}\bar{x}^i + \frac{n_0}{n+n_0}\mu_0, (n+n_0)\sigma^2\right) \tag{1.7}$$

$$\sigma^2 \mid x^i \sim \text{Ga}\left(\alpha + \frac{n}{2}, \gamma + \frac{1}{2}\sum_k (x_k^i - \bar{x}^i)^2 + \frac{nn_0}{2(n+n_0)}(\bar{x}^i - \mu_0)^2\right) \tag{1.8}$$

Algorithm 2 Thompson Sampling

for $t = 1 \cdots T$ **do**

Sample the variances $\sigma_{i,t}^2$ from their posterior according to 1.8

Sample the means $\mu_{i,t}$ according to 1.7

Solve (numerically) the following problem

$$\begin{aligned} \max_{\lambda} \quad & f_w(\lambda; \mu_t, \sigma_t^2) = w\sum_{i=1}^{K} \lambda_i \mu_{i,t} - (1-w)\frac{1}{K}\sum_{i=1}^{K}\frac{\sigma_{i,t}^2}{\lambda_i} \\ \text{s.t} \quad & \sum_{i=1}^{K} \lambda_i = 1 \ , \ \lambda_i \geq 0 \end{aligned} \tag{1.9}$$

The solution $\boldsymbol{\lambda}_t$ defines a probability distribution over the arms.

Draw $I_t \sim \boldsymbol{\lambda}_t$

Pull the arm I_t, observe $x_t^{I_t}$ and update the posterior distributions of arm i as in 1.7 and 1.8.

end for

where $\bar{x}^i$ is the average reward of arm i up to the current time step.

1.2.4 A UCB-like algorithm

An overview of the UCB algorithm

Here we give a short overview of the classical UCB-like algorithms for finite-time multiarmed bandit problem (for details refer to [4]). The goal is to share the intuition behind such strategies. Thus, for clarity, we limit the presentation to UCB-1 which is the simplest version of such methods.

Let's consider a K-armed bandit problem. Successive plays of arm i generate the sequence $X_{i,1}, X_{i,2}, X_{i,3}, ...$ which are i.i.d according to an unknown distribution with an unknown mean μ_i. These random observations (also called rewards) are independent across the arms as well, i.e $X_{i,t}$ and $X_{j,t'}$ are independent for $1 \leq i, j \leq K$ and $t, t' \geq 1$.

Such algorithms work by associating to each arm a quantity called *upper confidence bound* which is the sum of two terms. The first one is simply the empirical average reward of the arm. The second, also called the exploration term, is related to the width of a confidence interval. This results in a high probability upper bound, usually derived from Chernoff-Hoeffding concentration inequality. UCB-1 strategy

consists in playing at time t the arm i that maximizes $\mu_{i,t} + \sqrt{\frac{\alpha \log(t)}{T_{i,t}}}$, where $\mu_{i,t}$ is the average reward obtained from arm i. $T_{i,t}$ is the number of times arm i has been played so far. usually when the horizon is fixed the term $\log(t)$ may be replaced by $\log(T)$, where T is the time horizon.

Inspired by UCB's approach we suggest Algorithm 3 that we have called UCB-REGERR.

First steps towards UCB-REGERR

The first attempt was to use means and variances high probability bounds to derive an upper bound for the objective function. This means plugging upper bounds of the means and a lower bounds of the variances in the objective function of the problem 1.3 and solve it. The resulting $\boldsymbol{\lambda}$ is then considered as a distribution over the arms from which the next draw will be sampled. This version turned out to be unstable and thus not practical. Here follows a discussion of its weak points.

- **Why isn't it practical to use variances' lower bounds?** The problem is that the variance (high confidence) lower bound may be negative (when only few samples are available) which changes the problem from a concave one to a convex one. Thus the maximization problem doesn't make sense anymore.
- **Why is capping variances' negative bounds to 0 not a good idea?** A solution would have been to cap the the negative lower bounds to 0 until they go positive. The problem in doing so is that for a simple case of 2 arms where μ_1 and μ_2 are very different, say $\mu_1 > \mu_2$ (with a big gap), and similar variances $\sigma_1^2 \simeq \sigma_2^2$ the initial negative lower bound of σ_2^2 causes that the solution of the problem with plugged bounds is deterministic i.e $\boldsymbol{\lambda} = (1, 0)$. This is explained by the big gap between the means. Indeed, as soon as the upper bound of the second arm is of the order of μ_1 the solution would also recommend to draw arm 1 (as in classical UCB). This means that we keep on drawing the first arm without exploring the second one, the lower bound of which preserves its negative sign and the algorithm got stuck in the allocation $\boldsymbol{\lambda} = (1, 0)$.

To preserve the concavity of the problem, we switch from lower bound to upper bound of the variances. This final version is presented in Algorithm 3.

1.3 Empirical Analysis of UCB-REGERR the case of 2 gaussian arms

We first define a set of experiments on which we evaluate empirically UCB-REGERR. Table 1.1 details the parameters defining each experiment. Note that we distinguish 2 categories according to how balanced (i.e close to $\frac{1}{K} = 0.5$) the allocation λ is. The arms here are gaussian and thus defined by their mean and variance parameters.

Algorithm 3 UCB-REGERR

Parameters: n, w.

Initialization

Draw each arm twice to initialize the means $\hat{\mu}_i$ and variances $\widehat{\sigma_i^2}$ estimates. Define high probability upper bounds

$$\mu_i < B_{i,t}^{mean} \quad \sigma_i^2 < B_{i,t}^{var}$$

for $t = 1 \cdots n$ **do**

Solve (numerically) the following problem

$$\begin{aligned} \max_{\lambda} \quad & f_w(\lambda; B_t^{mean}, B_t^{var}) = \; w \sum_{i=1}^{K} \lambda_i B_{i,t}^{mean} - (1-w)\frac{1}{K}\sum_{i=1}^{K} \frac{B_{i,t}^{var}}{\lambda_i} \\ \text{s.t} \quad & \sum_{i=1}^{K} \lambda_i = 1 \;,\; \lambda_i \geq 0 \end{aligned} \tag{1.10}$$

The solution $\boldsymbol{\lambda}_t$ defines a probability distribution over the arms.

Draw $I_t \sim \boldsymbol{\lambda}_t$

Pull the arm I_t and update $\hat{\mu}_{I_t}, B_{I_t,t}^{mean}$ and $\widehat{\sigma_{I_t}^2}, B_{I_t,t}^{var}$

end for

1.3.1 Curse of unbalanced allocation for UCB-REGERR

UCB-REGERR is behaving better for "balanced" values of λ than for "unbalanced" ones. On Figure 1.4 (b) and (d), one can see that the allocation the algorithm suggests at each time step converges to the optimal one λ_{optim} for the balanced experiment °2, while for the unbalanced experiment °3, no convincing convergence can be stated. However, one may suspect a slow convergence from the shape of the curves.

To understand this behavior, let's focus on the experiment n°3. The optimal allocation for arm 1 is $\lambda_1 = \lambda = 0.18$. This means that when our draws allocation tends to the optimal one, we tend to pull arm 1 much less often than arm 2 (which is not the case for a balanced allocation λ). In this case, the upper bound of the small variance $\sigma_1 = 0.1$ happens to be higher than expected, because of the small number of draws of arm 1 which results in a high exploration term.

$$B_{1,t}^{var} = \widehat{\sigma_{1,t}^2} + \sqrt{\frac{\beta \log(n)}{T_{1,t}}}$$

This bound being higher than the actual value (0.1) makes the algorithm fall in a very different allocation especially when the actual value is very small. Indeed, according to Figure 1.2, $\lambda_{optim}(\sigma_1)$ varies faster when σ_1 is close to 0. Then, if we wish having a good approximation of λ_{optim}(=0.1 in this case), we should draw arm

experiment n^o	λ	μ	σ^2
1	0.53 (balanced)	(2,1)	(1,1)
2	0.54 (balanced)	(8,1)	(1,2)
3	0.18 (unbalanced)	(1.1, 1)	(0.1, 2)
4	0.77 (unbalanced)	(3,1)	(0.1,0.1)

Table 1.1: Experiments settings

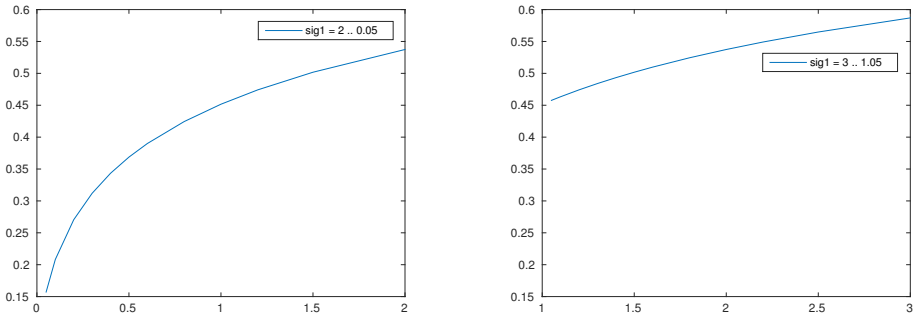


Figure 1.2: **Left:** Curve of λ_{optim} for $\sigma_1 \in [0.05, 2]$. **Right:** Curve of λ_{optim} for $\sigma_1 \in [1.05, 3]$

1 enough times so that the exploration term is very small, but this doesn't seem feasible since the closer we get to this small λ_1 the less we should pull arm 1.

To overcome this "slow convergence", we can suggest 2 solutions. The first one is a naive one that consist in ignoring the exploration term when the variance estimate is under a certain threshold. This is not surprising when we realize that plugging only the empirical estimates and not their upper bounds seems to provide the convergence already (see Figure 1.3). The second one is to consider Bernstein upper bounds that are used for UCB-V (see [3]). In this case the variance upper bound can be written as

$$B_{1,t}^{var} = \widehat{\sigma_{1,t}^2} + \sqrt{\widehat{\sigma_{1,t}^2}\frac{\beta \log(n)}{T_{1,t}}} + \eta\frac{\log(n)}{T_{1,t}}$$

This time the amplitude of the exploration term (1) is controlled by variance estimate directly which makes it vanish faster when the variances are very small. The exploration term (2) vanishes as $\frac{1}{t}$, which means faster than Chernoff-Hoeffding exploration term that converges only as $\mathcal{O}(\frac{1}{\sqrt{t}})$.

1.3.2 Regret Empirical Analysis

Here, we define a regret measure and analyze its behavior in some of the relevant experiment defined earlier.

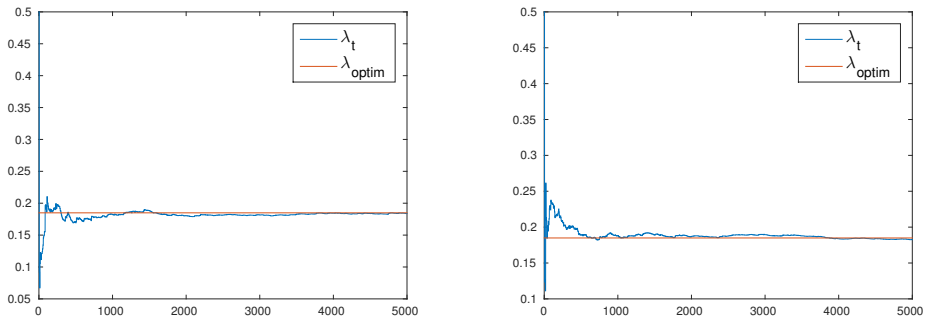


Figure 1.3: **Left:** Convergence of λ_t in the case of a unbalanced λ_{optim} (experiment n^o3) with a truncated variance upper bound (no exploration term for small variances). **Right:** Convergence of λ_t in the case of a balanced λ_{optim} (experiment n^o3) with empirical estimate instead of upper bounds

Let $\widetilde{\lambda}_t$ the draws allocation that the learning algorithm actually realized up to time step t, i.e for arm i the algorithm allocated $\widetilde{\lambda}_{i,t} = \frac{1}{t}\sum_{i=1}^{t} \mathbb{1}_{I_t=i}$. We measure the regret R_t at each time step t as the objective function gap between the optimal allocation λ_{optim} and $\widetilde{\lambda}_t$.

$$R_t = f_w(\lambda_{optim}) - f_w(\widetilde{\lambda}_t)$$

One may expect that R_t decays like $\frac{1}{\sqrt{t}}$. On Figure 1.4, we plotted the rescaled regret $\sqrt{t}R_t$ for experiments 2 and 3 when using the UCB-REGERR algorithm and the baseline algorithm ε-greedy, where $\varepsilon = 0.05$.

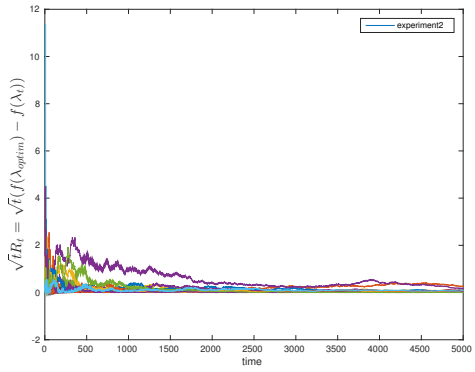
The results confirms the expected convergence rate of the regret. We can see on Figure 1.4 (c), which corresponds to a low convergence of λ, that the rescaled regret tends to some constant value which means that the regret converges as $\frac{1}{\sqrt{t}}$. However, when λ converges faster as in experiment 2, the rescaled regret looks like its converging to a very small constant. If in some cases, this constant is 0, the rate would be faster than $\frac{1}{\sqrt{t}}$, then one may consider these cases as the "easy" ones. This proves empirically that UCB-REGERR brings a solution to the regret-error trade off with an interesting convergence rate, which opens the way to the analysis of theoretical guarantees and consistency of the algorithm.

Discussion and future work

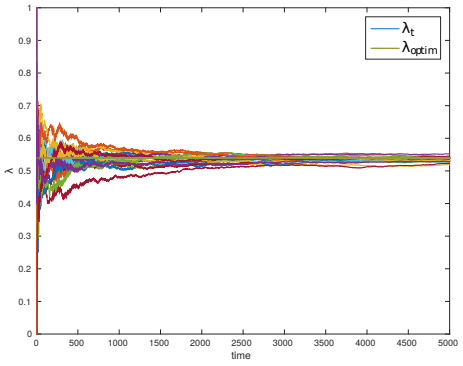
We have formulated a multiarmed problem that gathers two different subproblems that are usually studied and solved separately. This conciliation effort between two extreme problems happens to find applications in modern collaborative technologies such as educational platforms (MOOC) or crowdsourcing markets.

The empirical results stand as a strong justification of the algorithm as a promising track towards an interesting answer to the formulated problem. Our ambition now is to investigate more in the theoretical path to figure out more precisely the

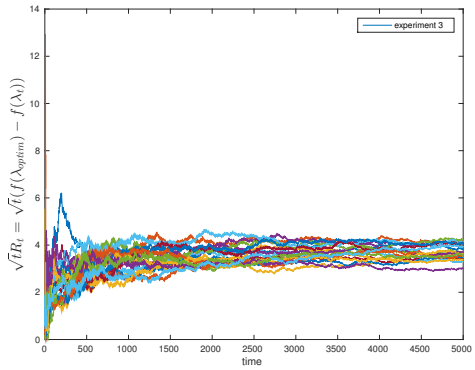
convergence rate under different conditions. Thus, the next steps will focus mainly on understanding the limits of the algorithm by designing few cases that constrain its performance, and eventually providing theoretical guarantees for the algorithm convergence and its regret rate.



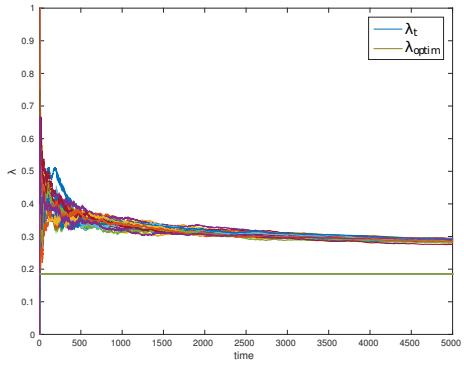
(a) Scaled regret, UCB-REGERR, exp $n^o 2$



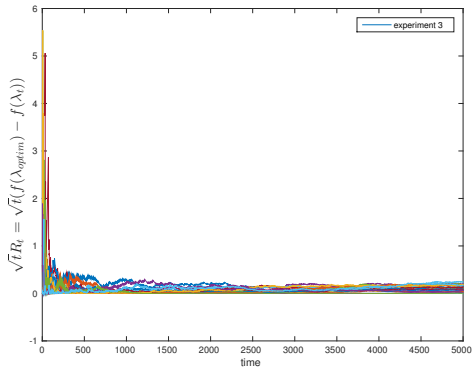
(b) $\widetilde{\lambda}_t$ convergence, UCB-REGERR, exp $n^o 2$



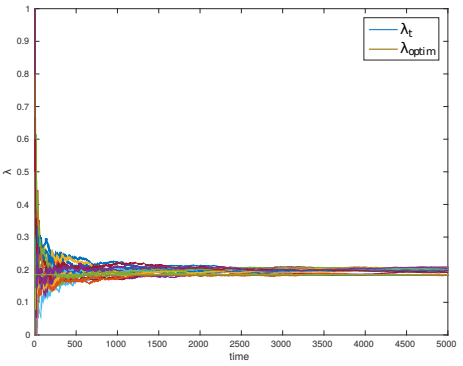
(c) Scaled regret,UCB-REGERR, exp $n^o 3$



(d) $\widetilde{\lambda}_t$ convergence, UCB-REGERR, exp $n^o 3$



(e) Scaled regret, 0.05-greedy, exp $n^o 3$



(f) $\widetilde{\lambda}_t$ convergence, 0.05-greedy, exp $n^o 3$

Figure 1.4: Scaled regret and $\widetilde{\lambda}_t$ convergence for UCB-REGERR and ε-greedy, 20 runs and $T = 5000$

Bibliography

[1] Andrs Antos, Varun Grover, and Csaba Szepesvri. Active learning in multi-armed bandits, 2009.

[2] Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. In *COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27-29, 2010*, pages 41–53, 2010.

[3] Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. Exploration-exploitation trade-off using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410:1876–1902, 2009.

[4] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2-3):235–256, May 2002.

[5] Alexandra Carpentier, Alessandro Lazaric, Mohammad Ghavamzadeh, Rémi Munos, and Peter Auer. Upper-Confidence-Bound Algorithms for Active Learning in Multi-Armed Bandits. In *ALT - the 22nd conference on Algorithmic Learning Theory*, Espoo, Finland, October 2011.

[6] Victor Gabillon, Mohammad Ghavamzadeh, Alessandro Lazaric, and Sébastien Bubeck. Multi-bandit best arm identification. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain.*, pages 2222–2230, 2011.

[7] Yun-En Liu, Travis Mandel, Emma Brunskill, and Zoran Popovic. Trading off scientific knowledge and user learning with multi-armed bandits, 2014.