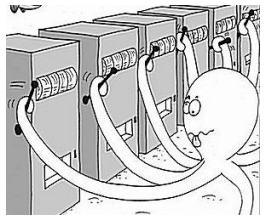
Sequential and reinforcement learning: Stochastic Optimization II



Summary

This session describes the important and nowadays framework of on-line learning and estimation. This kind of problem arises in bandit games (see below for details) and in optimization of big data problems that involve so massive data sets that the user cannot imagine exploiting the entire data with a batch algorithm. Instead, we are forced to use some subset of observations sequentially to produce both : tractable algorithms, fast predictions, efficient statistical estimators.

We will describe below some bandit algorithms for stochastic optimization when some important assumptions are made on the structure of the optimization problem. These famous algorithms address a typical situation where we need to adjust sequentially our prediction and our action according to online observations.

At last, we will briefly describe an ultimate method for solving the minimization of a non convex function with multiple local traps, by using simulated annealing. This problem will be illustrated on the so-called travelling salesman problem.

1 Bandit games

1.1 Motivation

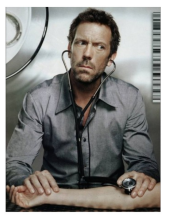
- You are in a casino and want to play with slot machines
- Each one can give you a potential gain, but these gains are not equivalent
- You *sequentially* play with one of the arms of the bandit machine

How to design a good policy to sequentially optimize the gain ?

This question is not so obvious because of the possibility you are offered to sequentially adjust your choice of the arm you will play at each round. This "toy" presentation can be indeed examplified in many concrete situations.

1.1.1 Clinical trials

Problem : Optimization of a sequence of clinical trials



Imagine you are a doctor :

- A sequence of patients visit you *sequentially* (one after another) for a given disease
- You choose one treatment/drug among (say) 5 availables
- The treatments are not equivalent
- You do not know where is the best drug, but you observe the effect of the prescribed treatment on each patient
- You expect to find the best drug despite some uncertainty on the effect of each treatment

How can we design a good sequence of clinical trials ?

1.1.2 "Fast fashion" retailer

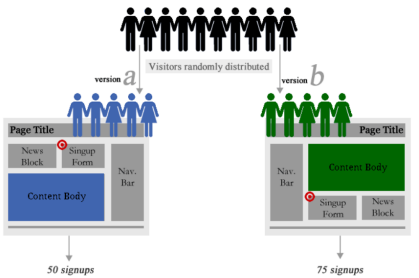
Imagine you are a firm solding clothes :


- A population of customers visit you *sequentially* each week/day
- You observe weekly/daily sales and measure item's popularity
- You want to restock popular items and weed out unpopular ones *on-line*
- You expect to maximize your benefit while finding the best items

How can we design a good sequence of fast-fashion operations ?

1.1.3 "Web design"

Imagine you want to select a web page design

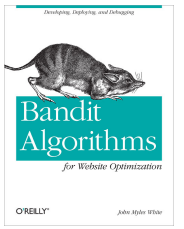


- A population of customers visit you *sequentially* (one after another)
- You randomly propose two designs a and b and measure design's popularity through the signups you obtain
- You want to propose the popular design to maximize your benefit

How can we build a good sequence of webpage propositions ?

1.1.4 Other motivating examples

- Pricing a product with uncertain demand to maximize revenue
- Trading (sequentially allocate a ratio of fund to the more efficient trader)
- Recommender systems :
  - advertisement
  - website optimization
  - news, blog posts



- Computer experiments
  - A code can be simulated in order to optimize a criterion
  - This simulation depends on a set of parameters
  - Simulation is costly and only few choices of parameters are possible

1.2 Why is this problem difficult ?

Needs a careful mathematical formalization :

What is the underlying optimization problem to be solved ?

The mathematical model probably involves both

- one exploration step (the problem of exploring many possible arms to locate the optimal one)
- one exploitation step (the problem of playing as much as possible the best possible arm)

A trade-off certainly exists between we will have only a finite number of plays (say T in what follows). Hence, we have to manage both views :

Scientist : Explore new ideas / Businessman : Exploit best idea found so far

1.3 Mathematical definition

Environment :

- At your disposal : d arms with *unknown* parameters $\theta_1, \dots, \theta_d$.
- For any time t, your choice is described by one action $I_t \in \{1 \dots, d\}$
- For any time t, you receive a reward, that depends on your choice I_t :

$$A_t^{I_t}$$

For example :
— it corresponds to the money obtained by sampling one specific slot machine in a Casino, the number of the machine is I_t.
— it corresponds to the size of a tumor after choosing to test one drug on a patient.

Reward distribution :

- Of course, the rewards cannot be reasonnably assumed to be deterministic (otherwise I won't be there to talk about it !)
- For a fixed choice of the arm i, the rewards are i.i.d.

$$(A_t^i)_{t\geq 0} \sim \nu_{\theta_i}.$$

- Important assumption : the reward distribution ν_θ belongs to a parametric family of probability distributions (Exponential, Poisson, ...).
- One typical example : the reward of arm i is distributed according to a Bernoulli distribution $\mathcal{B}(\theta_i)$:

$$\mathbb{P}[A_t^i = 1] = \theta_i \qquad \text{and} \qquad \mathbb{P}[A_t^i = 0] = 1 - \theta_i.$$

Expected reward : Since the reward of arm i are assumed i.i.d., the expected reward of arm i is given by

$$\forall t \in \mathbb{N} \qquad \mathbb{E}A_t^i = \mathbb{E}A^i$$

1.4 A simplification assumption

In what follows, we will restrict our lecture to the simplest case of Bernoulli rewards $\nu_p = \mathcal{B}(p)$:

- you obtain a gain of 1 with probability p
- 0 otherwise (with probability $1 - p$).

We use now p instead of θ to refer to the unknown parameters.

What is unknown, the several probability of success : $(p_1, \dots, p_d)$.

Without l.o.g., we assume that the first arm is the best one :

$$p_1 > \max_{2\leq j\leq d} p_j.$$

Admissible policy :

- The agent's action follow a dynamical strategy, which is defined on-line :

$$I_t = \pi\left(A_{t-1}^{I_{t-1}} \dots, A_1^{I_1}\right).$$

It means that at step t, we can use all the informations gathered from time 1 to time $t-1$ to make our decision I_t.
- The decision I_t can be driven either by
— a deterministic function
— a random function
of the information from 1 to $t-1$.

Final goal : Maximize (in expectation) the cumulative rewards :

$$\mathbb{E}\left[\sum_{t=1}^{n} A_t^{I_t}\right].$$

1.5 Regret of Bandit algorithms

We introduce the so-called cumulative pseudo-regret of any Bandit policy that compares the average performance of the algorithm used with the one if the optimal arm were known :

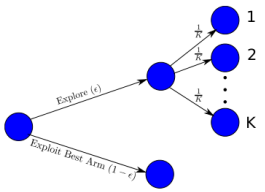
$$\bar{R}_n := \max_{1\leq j\leq d} \mathbb{E}\left[\sum_{t=1}^{n}(A_t^j - A_t^{I_t})\right] = p_1 n - \mathbb{E}\left[\sum_{t=1}^{n} A_t^{I_t}\right],$$

Minimizing $\bar{R}_n$ is equivalent to maximizing the average cumulative rewards of the policy over the n first runs of the algorithm.

2 ϵ-greedy algorithm

2.1 Description of the algorithm

Widely used **ϵ-greedy algorithm**'98 :



- Consider $\epsilon > 0$ and an initial guess of the ability of each arm :

$$\hat{p}_j(0) \qquad \text{is a prior information on} \qquad p_j$$

  If no information, sample p_j at random for example at the initialization step.
- Step t to $t+1$:
  — With probability $1-\epsilon$, use (one of) the best arm according to the belief you have at time t :

$$(\hat{p}_j(t))_{1\leq j\leq d}.$$

  — With probability ϵ/d, pick an arm uniformly among all possibles.
  — Upgrade the estimators of the Bernoulli parameters with the empirical means :

$$\hat{p}_j(t+1) = \frac{1}{n_j(t+1)} \sum_{n=1}^{t+1} A_j^n \mathbf{1}_{\text{arm } j \text{ sampled at time } n},$$

  where $n_j(t+1)$ is the number of times arm j is used from 1 to $t+1$.
- Usually, $\epsilon = 0.1$.

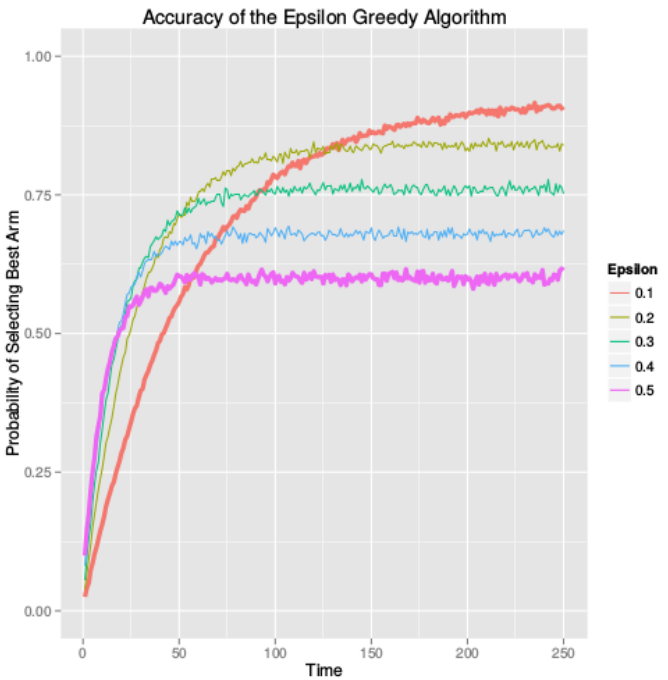
2.2 Pro and cons

Some interesting features of this method :

- Maybe the most simple algorithm to understand
- Maybe the most simple algorithm to program
- Maybe one of the most fastest algorithm to build a recommandation at each step

But unfortunately :

A brief experiment with 5 Bernoulli reward probabilities : $[0.1, 0.1, 0.1, 0.1, 0.9]$



- $\epsilon = 0.1$: Businessman and Learns slowly and Does well at the end
- $\epsilon = 0.5$: Scientist and Learns quickly but Does not exploit at the end

Whatever ϵ is, linear regret with n since the probability to select the best arm does not tend to 1. This probability is indeed $(d-1)/d \times \epsilon > 0$. The regret is at the least

$$\bar{R}_n \geq \epsilon \frac{d-1}{d} \Delta \times n,$$

where

$$\Delta = p_1 - \max_{j \geq 2} p_j.$$

2.3 ϵ-greedy improvement

The idea is to adjust the behaviour of the algorithm with a suitable annealing on ϵ all along the iterations of the algorithm. It can be shown (see [7]) that a suitable scheme is :

$$\forall n \geq 1 \qquad \epsilon_n = 1 \wedge \left\{ c \frac{d}{\Delta^2 n} \right\},$$

where c is a non-negative constant and Δ is an unknown parameters that involves the difference ability between the two best optimal arms :

$$\Delta = p_1 - \max_{j \geq 2} p_j.$$

In practice, Δ is unknown but may be estimated on-line. It can be easily shown that in that case

$$\bar{R}_n \sim \Delta^{-2} \log n,$$

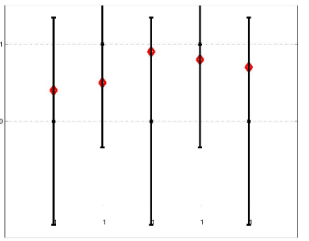
which is the optimal rate that can be achieved when the distributions do not change with n (see [5] for the optimality of this bound).

3 Optimistic algorithms

3.1 Rough description

"Optimal" algorithms may be produced by using an optimistic approach. These optimistic algorithms are the so-called *Upper Confidence Bound* methods (UCB for short). They have been introduced in 1985 in [5]. One of the recent advances on these methods may be found in [2].

We propose a description of the general principle of the algorithm.



Strategy :

- Build a confidence bound around each empirical estimation of the probability of success

$$\hat{p}_i(t) \in [l_i(t); u_i(t)], \forall 1 \leq i \leq d$$

- at time t, select the arm with the highest upper confidence bound :

$$I_t = \arg\max u_i(t).$$

- Get the reward, and update the empirical estimator and the confidence bounds

$$\hat{p}_i(t+1) \in [l_i(t+1); u_i(t+1)]$$

3.2 Pro and Cons

UCB-like algorithm are shown to be optimal and satisfy

$$\limsup_{n \longrightarrow +\infty} \frac{\bar{R}_n}{\log n} \leq \sum_{p < p_1} \frac{1}{2(p_1 - p)},$$

when the distributions of success are kept fixed with the horizon n. Hence, they are (minimax) optimal with respect to the lower bound on all possible strategies (see [5]).

Moreover, they also achieve a distribution free upper bound on the regret which is also optimal (see [1]).

$$\forall (p_1, \ldots, p_d) \in [0,1]^d \qquad \bar{R}_n \lesssim \sqrt{d \log(d) n}$$

4 Narendra Shapiro algorithm

4.1 Initial algorithm (1969)

The so-called Narendra-Shapiro bandit algorithm (NSa for short) was introduced in [6]. It defines a probability vector of $\mathcal{S}_d$

$$X_t = (X_t^1, \ldots, X_t^d) \quad | \quad \sum_{j=1}^{d} X_t^j = 1.$$

Idea : Use X_t to sample one arm at step t and then upgrade this probability X_t.

4.1.1 Two arm NSa

- In the two-armed situation with $p_2 < p_1$, denote $X_t = (x_t, 1 - x_t)$
- $X_t(1) = x_t$ is the probability to choose the first arm at step t.
- $X_t(2) = 1 - x_t$ is the probability to choose the second arm at step t.
- Upgrade formula

$$x_{t+1} = x_t + \begin{cases} \gamma_{t+1}(1 - x_t) & \text{if arm 1 is selected and wins} \\ -\gamma_{t+1} x_t & \text{if arm 2 is selected and wins} \\ 0 & \text{otherwise} \end{cases}$$

- Common step size :

$$\gamma_t = (1 + t/C)^{-\alpha}, \qquad \alpha \in (0, 1) \qquad \text{with large enough } C.$$

- Same idea :
  — If you win : reinforce the probability to sample I_t w.r.t. the remaining weights $(X_t^j)_{j \neq I_t}$ and decrease the probability to sample the other arms accordingly.
  — If you loose $(A_t^{I_t} = 0)$: do nothing.

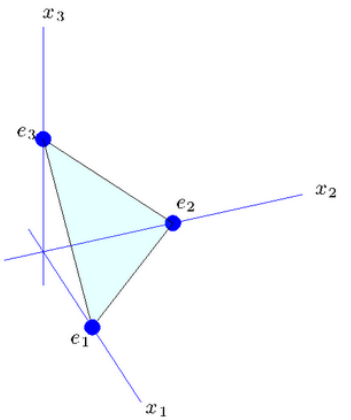
4.1.2 Multi-armed arm NSa

Multi-armed situation, I_t : arm sampled at time t, $A_t^{I_t}$: obtained reward. Upgrade

$$\forall j \in \{1 \ldots d\} \qquad X_t^j = X_{t-1}^j + \gamma_t \left[\mathbf{1}_{\{I_t = j\}} - X_{t-1}^j \right] A_t^{I_t}$$

To sum up :

- If you win : reinforce the probability to sample I_t and decrease the probability of others.
- If you loose $(A_t^{I_t} = 0)$: do nothing.



Unfortunately, this algorithm does not perform very well from the regret viewpoint : it has a strictly positive probability to converge towards a wrongly optimal arm, leading to a linear regret.

4.2 Two armed over-penalized NSa (2018)

To bypass the difficulty of convergence towards a wrong target, we increase the exploration of the algorithm by adding a penalty effect to reinforce the escape from local traps (see [3]).

5 Homework project

5.1 Feature of a good homework

Length limitation : 20 pages !
Deadline : 25th of March.
Group of 2 students allowed.

- This report should be not too long : strictly less than 20 pages, including the references.
- The work relies both on a numerical implementation part (with either R, Matlab or Python) and on a description of the algorithms behaviour on your report, with experiments and theoretical insights.

The report should be organized as follows

1. First motivate the problem with a concrete application and propose a reasonable modelling.
2. Second, the report should explain the mathematical difficulties to solve the model and some recent developments to bypass these difficulties. You can also describe the behaviour of some algorithms.
3. Third, the report should propose :
   - numerical simulations using packages found on the www or your own experiments.
   - some sketch of proofs of baseline theoretical results
   - a discussion part that present alternative methods (with references), exposing pros and cons of each methods.

You can choose to only exploit a subsample of the proposed references, as soon as the content of your work is interesing enough. You can also complement your report with a reproducible set of simulations (use R, Matlab or Python please) that can be inspired from existing packages. (If packages are not public, send the whole source files). These simulations are not accounted in the 20 pages of the report.

The report files should be named lastname.doc or lastname.pdf and expected in my mailbox before 25th of March.

And to do this, anything is fair game (you can do what you want and find sources everywhere, but take care to avoid a plagiat !)

5.2 Roadmap of your study and report

Section 1

- Explain and motivate the problem.
- Find a concrete example and a real database on www.

Section 2

- Fill the missing details of the algorithms briefly described in these slides.
- Explain the theoretical/numerical advantages of the methods.
- What is the Lai and Robbins lower bound result ?
- Find and describe another algorithm (that is not shown here).

Section 3

- Implement the previous algorithms with your own scripts (don t use any package) on situations where you receive parametric rewards in typical families (Bernoulli, Exponential, . . .). The implementation is generally easy.
- Comment on the several features of the algorithms you implement.
- Handle the real dataset example. This time you can use packages !

Références

[1] Jean Yves Audibert et Sébastien Bubeck, *Regret Bounds and Minimax Policies under Partial Monitoring*, Journal of Machine Learning Research **11** (2010), 2635–2686.

[2] O. Cappé, A. Garivier, O. Maillard, R. Munos et G. Stoltz, *Kullback-Leibler Upper Confidence Bounds for Optimal Sequential Allocation*, Annals of Statistics **41** (2013), 1516–1 541.

[3] S. Gadat, F. Panloup et S. Saadane, *Regret bound for Narendra-Shapiro bandit algorithms*, Stochastic Processes and Applications (2016).

[4] Gittins J.C., *Bandit Processes and Dynamic Allocation Indices*, Journal of the Royal Statistical Society. Series B (Methodological) **41** (1979), 148–177.

[5] T.L. Lai et H. Robbins, *Asymptotically efficient adaptive allocation rules*, Advances in Applied Mathematics **6** (1985), 4–22.

[6] K. S. Narendra et I. J. Shapiro, *Use of stochastic automata for parameter self-optimization with multi-modal perfomance criteria*, IEEE Trans. Syst. Sci. Cybern. **5** (1969), 352–360.

[7] Auer P., Cesa Bianchi N. et Fischer P., *Finite-time Analysis of the Multiarmed Bandit Problem*, Machine Learning **47** (2002), 235 ?256.