Work in Lri

Xinneng XU

19 juillet 2019

Table des matières

1 25/06/2019 :

1.1 DQN batch

Coder DQN With Batch

I created a new version of DQN which trains the network with batch, rather than train the network with each simulation.

What it was :

Loop :

- simulate one time and get tuple $<s_t, r_t, s_{t+1}>$
- set $y_t = r_t + \beta \min_a Q_t(s_{t+1}, a; w_t)$
- train the network by minimizing the loss with the input $= X = (F(s_t), G(a)), Y = y_t$

What I modified :

Loop :

- there is memory D and I simulate $n = (20)$ times, save the each tuple $<s_t, r_t, s_{t+1}>$ in D
- sample a batch(=10) from D and calculate y for each tuple in the batch
- train the network by minimizing the loss of the batch

Comparison. With Time=2000(each time correspond to one simulation),K=5, and the cost of arm a is calculated by formula : $c(a) = 10 * (a + 1) + 55 * theta\_true[a]$ where theta_true = [0.9, 0.64013, 0.50242, 0.37156, 0.26535].

The regret/time of DQN without batch and UCB is 1.988 and 3.617 respectively.

Unfortunately, the DQN training with batch, it doesn't work. The problem is that when I trained the network, after several epochs, the output of network will be -inf. I have tried to use *Keras* to build the network and use *mean square error* as loss function, it doesn't work, the problem is when I trained the network, after several epochs, the output of network will be the same for any input.

2 Before 02/07/2019 :

2.1 DQN with(out) batch and Experimentation

Before 01/07

Before today, I have implemented Deep Q-Learning algorithm training with(out) batch, simple UCB. And I have gotten the results of executing UCB for 20 epochs, 30000 iterations, 100 resources. I have given a presentation of my work in Nokia.

01/07 - 02/07

- Rebuilt the code to separate the part of data and algorithm
- Added comments for every variable and function
- Wrote the manual for others can use my programme easily.

3 03/07/2019 :

3.1 DQN batch

Results of Experimentation for Ext with T=6000

The calculate of executing algorithm extension with T=6000, epochs=20, 100 resource has been finished, But the result of regret per T Figure 1 is very strange, I am executing on epoch more to find whether the result of regret is always strange or not.

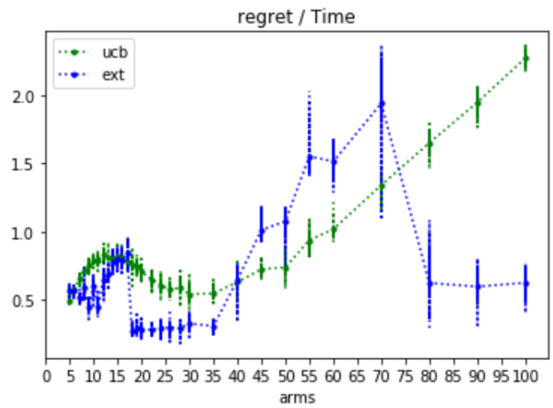


Figure 1 – Comparison of Ext and UCB with T=6000 epochs=20

Modified the Code and Re-execute the Program

I modified the number of T of every test(extract 10 resources) when I execute algorithm, and it is calculating.

Understand the Program of Stephan

I have understand the code of algorithms of LSE, LSE-BACKTRACK and how to execute the algorithms.

4 04/07/2019 :

4.1 Meeting in LINCS

Work in Nokia

Today, I have a meeting with Lorenzo and Johanne in LINCS, and we have discussed the graphs I should do :

- For artificial data, modify the 4 functions of simulation for the optimal x isn't always the same(0.5) and make it as a parameter.
- Execute the algorithms for artificial data and design graphs of regret and approximation.

5 05/07/2019 :

5.1 Experimental Work

Execute Algorithm and Draw Graphs

I have executed algorithms : ['lse', 'lse-backtrack', 'lse-weighted', 'without gradient', 'sgd'] using 3 sampling functions (Figure 2) : ['quadra', 'triangle', 'shark'] with $X_{opt} = 0.5$, Time=10000.

When I executed sgd, there was a problem and I implemented a classic sgd by sampling n times $x - d$ and get the mean reward r_{x-d} and sampling n times $x + d$ and get the mean reward r_{x+d}, then get $x_{new} = x + lr * \frac{r_{x+d} - r_{x-d}}{2*d}$.

— Sampling Functions.
X-axis represents the arms which are continuous from 0 to 1 and Y-axis represents the reward corresponding to each arm. The reward of X_{opt} is 1.

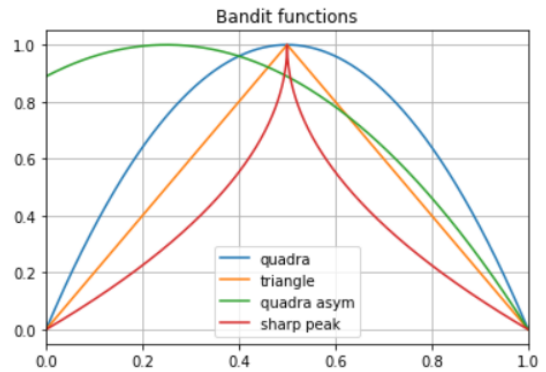


FIGURE 2 – 4 Sampling Functions

— Regret.
I have just executed the algorithms 1 time while X-axis represents time(each time sample 1 arm) and Y-axis represents cumulative regret.

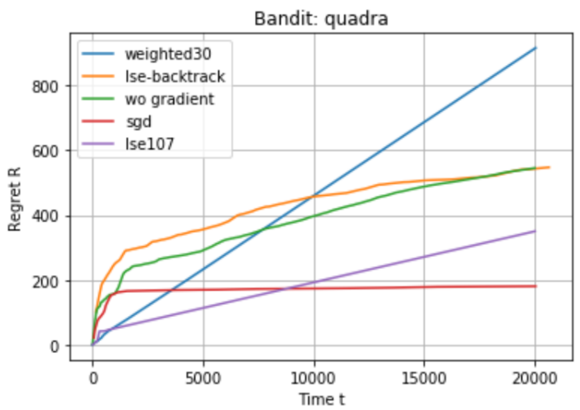


FIGURE 3 – 4 Sampling Functions

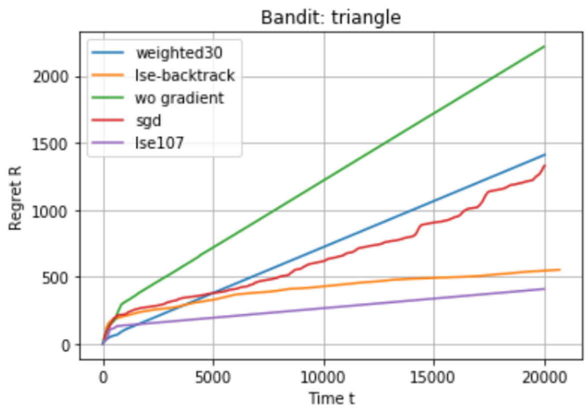


FIGURE 4 – 4 Sampling Functions

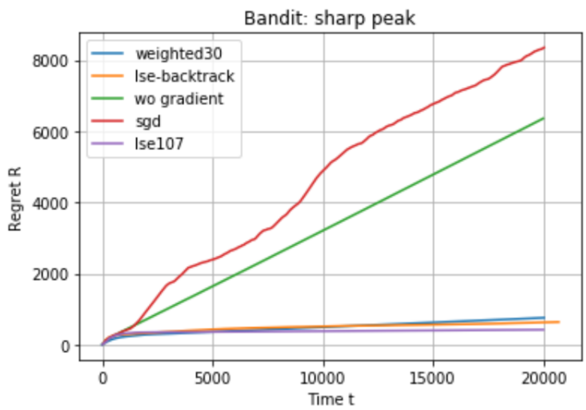


FIGURE 5 – 4 Sampling Functions

— Approximation.
I have just executed the algorithms 1 time while X-axis represents time(each time sample 1 arm) and Y-axis represents $abs(x - X_{opt}$ where x is the arm sampled.

Problems

We can see the graphs to get that 'lse-backtrack' is worse than 'lse' which is unreasonable.

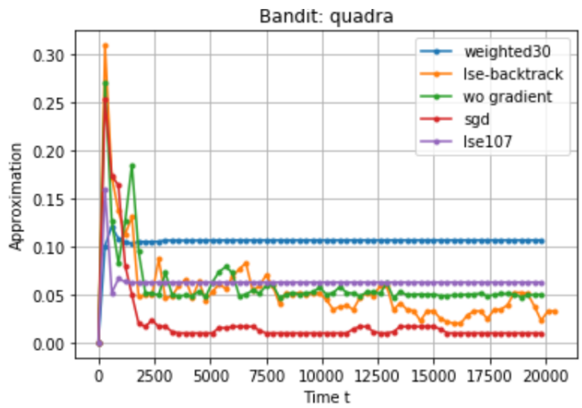


Figure 6 – 4 Sampling Functions

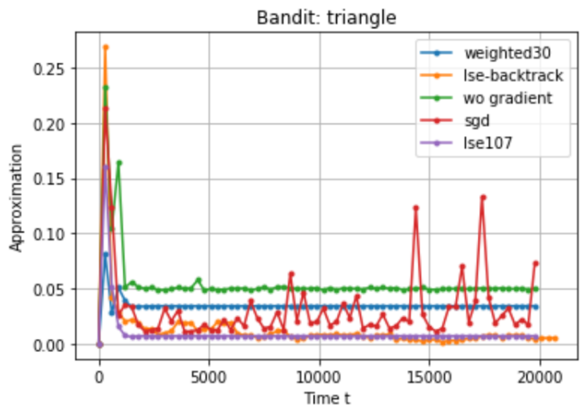


Figure 7 – 4 Sampling Functions

6 08/07/2019 :

6.1 Experimental Work(mean)

Solve Problems

— le 05/07 I encountered a problem which is that 'lse' is better than 'lse-backtrack', the reason is that the number of sampling isn't the same for the two algorithms, and for 'lse-backtrack' there are 6 points which needs more sampling for identify which point get max value.
So I tried the different number of sample for 'lse' and 'lse-backtrack'.

— For each algorithm, I execute N(=30) times and calculate the mean to increase accuracy of results.

Results

I executed algorithms for T=20000, x_{opt} = 0.5 and N = 30.

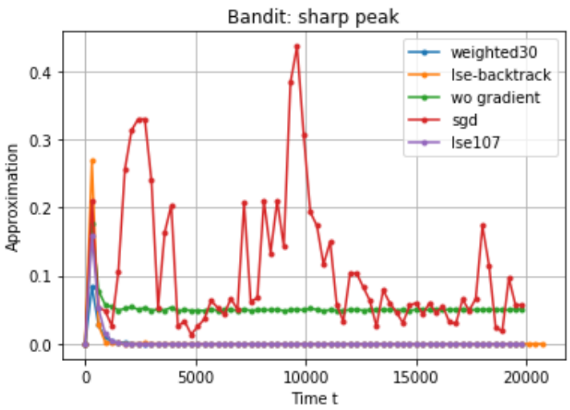


FIGURE 8 – 4 Sampling Functions

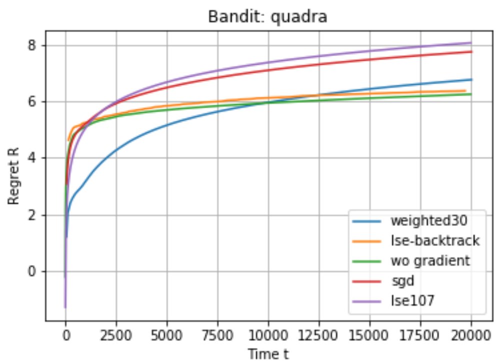


FIGURE 9 – 4 Sampling Functions

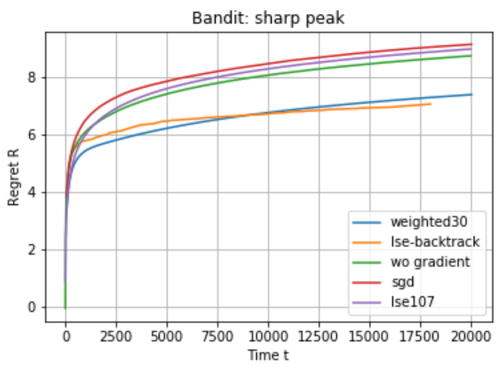


FIGURE 10 – 4 Sampling Functions

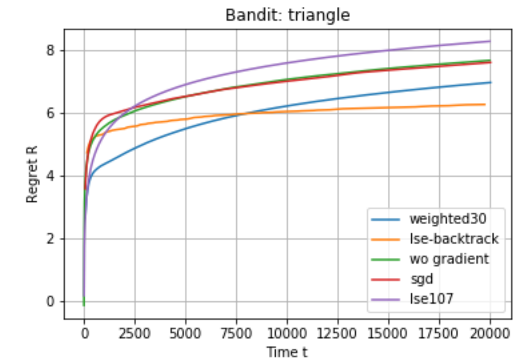


FIGURE 11 – 4 Sampling Functions

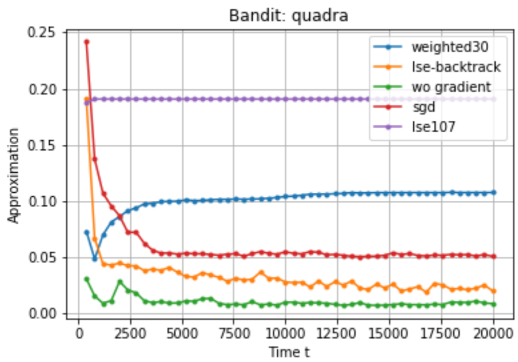


FIGURE 12 – 4 Sampling Functions

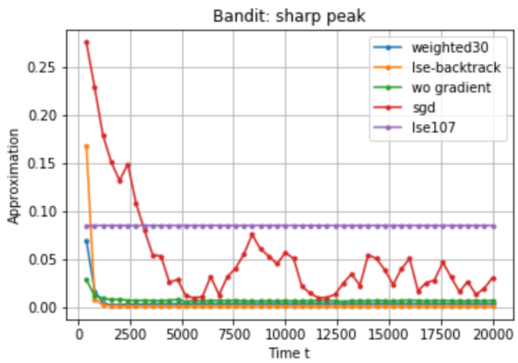


FIGURE 13 – 4 Sampling Functions

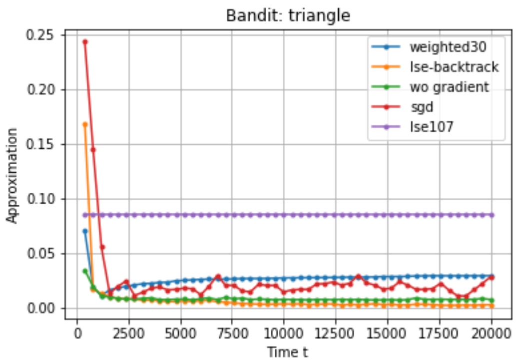


FIGURE 14 – 4 Sampling Functions

7 15/07/2019 :

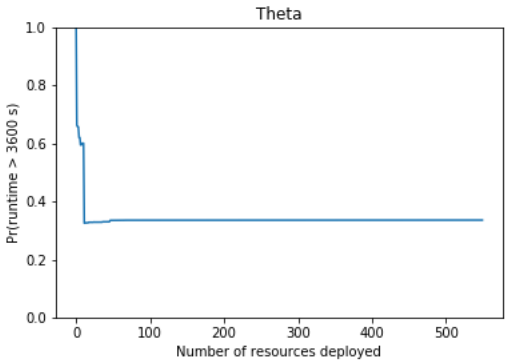
7.1 Start Semi-reel Data

Study on Semi-real Data

There are 5 types of semi-real data proposed by Stephan, but it is necessary to find what represents arm x and what represents QoS.

5 types :

— Parallel Workloads Archive
  However, the data isn't suitable for us ??



— Google Cluster Data
  This one is suitable for us Figure 15

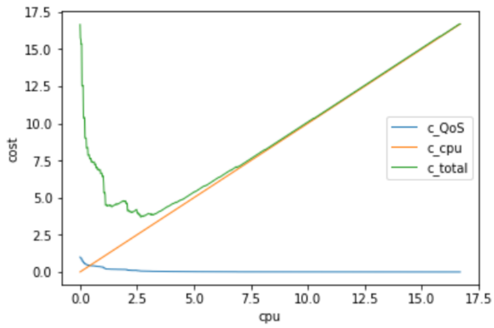


Figure 15 – Cost Google Cluster Data

— Riiser : mobile http streaming scenarios
  This one is suitable for us Figure 16
— Call tests measurements
— Makram

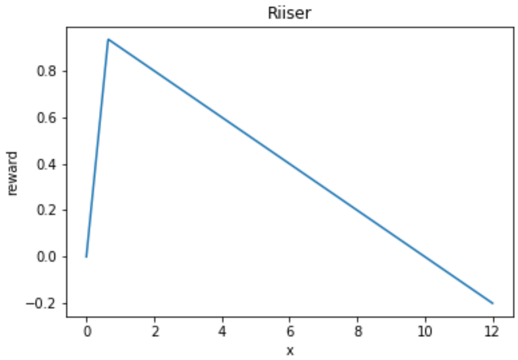


FIGURE 16 – Cost Google Cluster Data

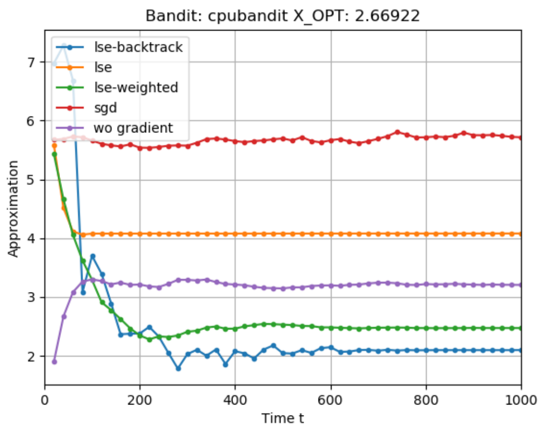
8 16/07/2019 :

8.1 Curves Semi-reel Data

Semi-real Data

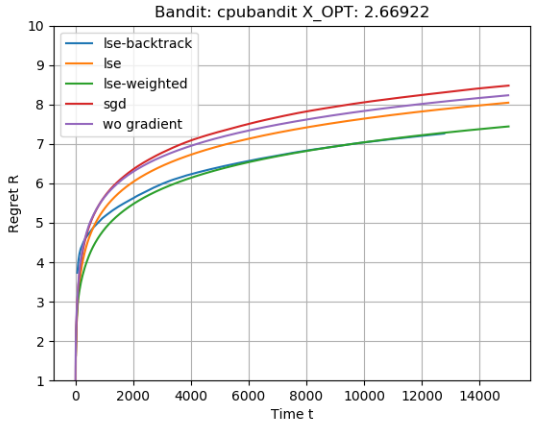
I executed the algorithms on semi-real data for T = 15000, rounds = 15, number of sampling = 9 :

— Approximation :



We can get that LSE-backtrack and LSE-weighted are much better than LSE, but they converge more slowlyl.

— Regret :



9 17/07/2019 :

9.1 X optimal outside Interval initial

Execute LSE-BACKTRACK with different intervals

I executed algorithm 'LSE-BACKTRACK' with parameters :

— curve 'Sharp Peak'

— $X_{opt} = 0.4$

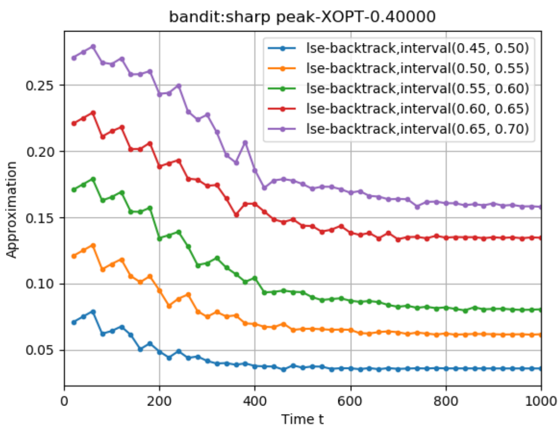
— T = 10000

— epochs = 80 (execute algo 80 times and then calculate mean value)

— number of sampling = 10

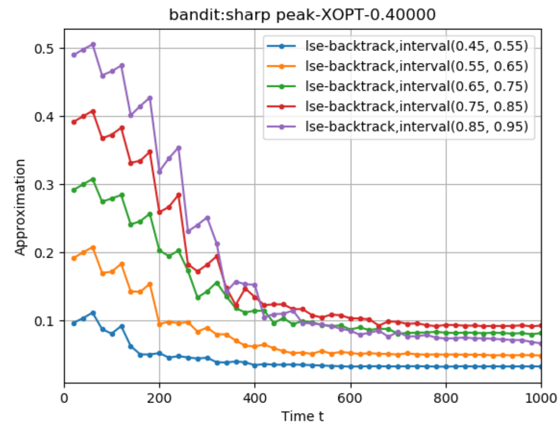
For intervals initial :

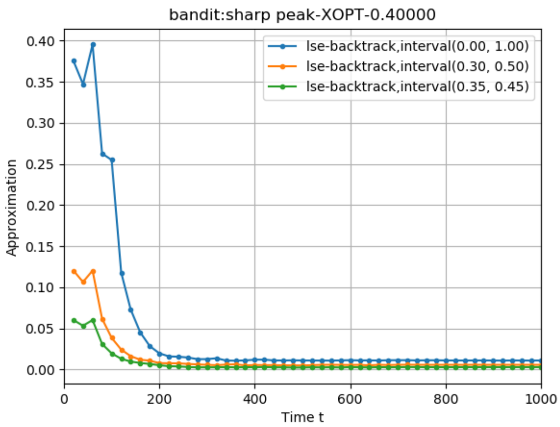
— intervals = [(0.45,0.5), (0.5,0.55), (0.55,0.60), (0.6,0.65),(0.65,0.7)] : Figure ??



— intervals = [(0.45,0.55), (0.55,0.65), (0.65,0.75), (0.75,0.85),(0.85,0.95)] :Figure ??

— intervals = [(0,1),(0.3,0.5),(0.35,0.45)] Figure : ??





9.2 For Article

Sub Conclusion

For Artificial Data

- There are 3 types of functions for representing theta : 'quadra', 'sharp peak', 'triangle' : Figure 17.
- When algo samples arm x, we use Normal distribution to add noise in the reward return to algo : Norm(mean : $theta(x)$, standard deviation : $0.2 + theta(x)/1.2$) where $theta(x)$ is calculated by function in Figure 17.
- For LSE, LSE-backtrack, LSE-weighted, we should define number of sampling(NS). We have executed algorithms by setting NS from 8 to 20.
- To avoid contingency, we have use rounds from 10 to 80(ex. execute 80 times for each algorithm and calculate the mean value).
- For function 'Sharp Peak', $X_opt = 0.4$, T = 15000, rounds = 80, number of sampling = 10, the curves of approximation and regret :Figure 20 21
- Intervals. To test the performance of LSE-backtrack, it is necessary to set the interval initial (x_i, x_j) where $x_i < x_j \wedge x_{opt} \notin [x_i, x_j]$

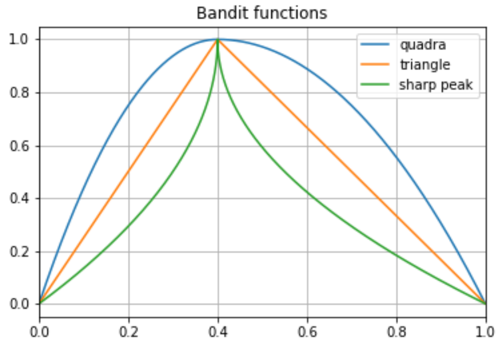


FIGURE 17 – Bandit functions XOPT=0.4

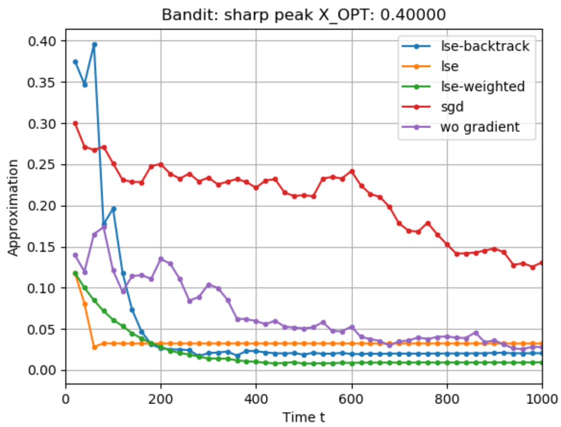


FIGURE 18 – Approximation XOPT = 0.4

10 18/07/2019 :

10.1 Work on Paper

Modify LSE-weighted in Paper

Execute for Google cluster data

I have executed algorithms for Google cluster data for more rounds to achieve that the

11 19/07/2019 :

11.1 Work in LINCS

— Meeting with Lorenzo.

— Try more different intervals for LSE-backtrack. I have tried :

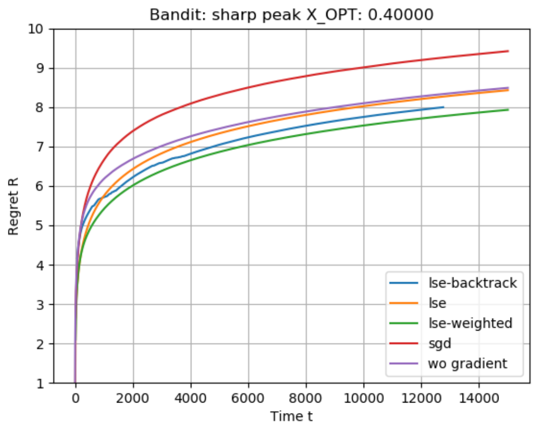


FIGURE 19 – Regret XOPT=0.4

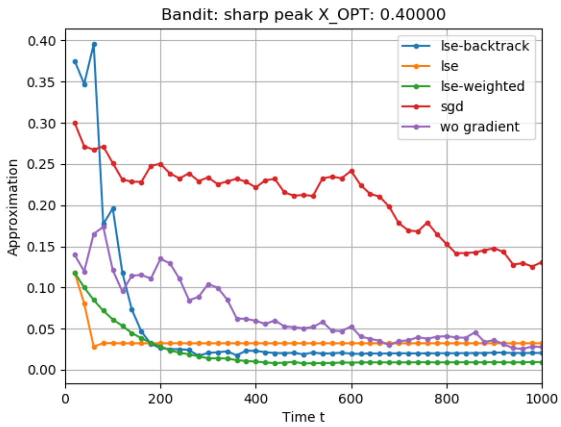


FIGURE 20 – Approximation XOPT = 0.4

intervals1 = [(0.4,0.45),(0.45,0.5),(0.5,0.55),(0.55,0.6),(0.6,0.65)]
intervals2 = [(0.45,0.55), (0.55,0.65), (0.65,0.75), (0.75,0.85),(0.85,0.95)]
intervals3 = [(0.45,0.65),(0.55,0.75),(0.65,0.85),(0.75,0.95)]
intervals4 = [(0.45,0.75),(0.5,0.8),(0.55,0.85),(0.6,0.9),(0.65,0.95)]
intervals5 = [(0.45,0.85),(0.5,0.9),(0.55,0.95)]

— For a interval, try different number of simple for LSE-backtrack. I have tried number of simple : $10, 20, \ldots, 100$

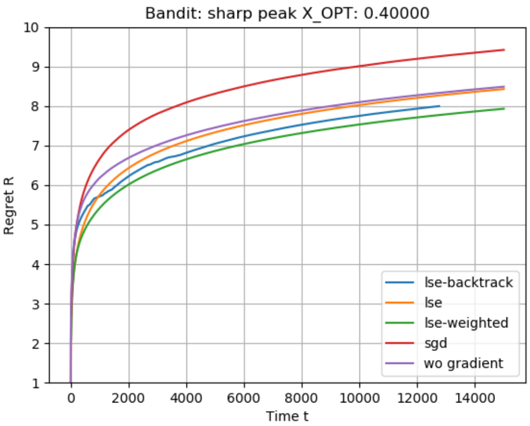


FIGURE 21 – Regret XOPT=0.4