Bandits via Gradient Optimization

Paul MARTIN - Samuel DIAI

February 5, 2021

1 Introduction

The aim of the project is to define a new algorithm for linear multi-armed bandit problems using gradient descent. In the course, we saw that UCB algorithm can be adapted to linear bandit but at each time step we need to estimate the weight vector by inverting a matrix of order $K \times K$, with K the number of arm considered. In many real application such as movie recommendation system, each arm represent a movie therefore the number of arm may be very important. The matrix inversion has typically a complexity $\mathcal{O}(K^3)$ (or $\mathcal{O}(K^2)$ by using Sherman-Morrison formula) which might be prohibitive to compute at each time step. In order to avoid this costly approach we use an online gradient descent updating the weight at each time step. In bandits algorithm, we need to take into account uncertainty and since we don't have access to concentration inequality with gradient descent we use an adapted softmax exploration called Boltzmann-Gumbel exploration. The perturbations are scaled in function of the quality of the mean reward estimator contrary to the classical softmax exploration. In this project we created an algorithm called BGE-SGD using Boltzmann-Gumbel exploration and stochastic gradient descent.

2 Problem definition

2.1 Linear Bandits

We focus on the linear bandits problem (we could also generalized to generalized linear model). Let's denote K the number of arm. Each action a lead to a reward $r(a)$ which follows an unknown distribution with mean $\mathbb{E}(r(a))$ bounded. We suppose that each action is characterized by a vector of features denoted $\phi(a)$. In the linear bandit case, if we choose the arm a at time t, we suppose that the reward has the following form :

$$r_t(a) = \phi(a)^T \theta^\star + \epsilon_t \quad \text{with} \quad \mathbb{E}(\epsilon_t) = 0$$

The agent observes noisy observations of the reward. The goal of the agent is to maximize the expected sum of reward (here undiscounted) up to the game horizon N.

$$\min \mathbb{E}(\sum_{t=1}^{N} r_t)$$

We can define the regret of the algorithm, i.e. the difference between what the agent has earned and what it could has earned at most by choosing the best action (in average).

$$R_t = \mathbb{E}(\sum_{t=1}^{N} r_t(a_t)) - \max_a \mathbb{E}(\sum_{t=1}^{N} r_t(a))$$

2.2 Least Square Estimate

At each step of the algorithm, we can estimate β by doing a linear regression. We also add a regularization term (RIDGE regression).

$$\theta_t = \underset{\theta}{\operatorname{argmin}} \sum_{s=1}^{t} (r_s - \phi(a_s)^T \theta)^2 + \lambda \|\theta\|_2^2$$

This formulation has a closed form solution :

$$A_t = \sum_{s=1}^{t} \phi(a_s)\phi(a_s)^T + \lambda 1_D \quad b_t = \sum_{s=1}^{t} \phi(a_s) r_s$$

$$\hat{\theta}_t = A_t^{-1} b_t$$

The drawback of this estimation is that at each step, the algorithm needs to inverse a matrix which can have a high dimension if the number of arm is important. For example it's often the case for a recommender system since each arm represents a proposition (a movie for example). The aim of the project is to avoid this costly operation by doing an online gradient descent estimation.

2.3 Linear UCB

In the previous section, we saw how to obtain an estimator of the reward average. In the Bandits algorithms we combine this approximation with uncertainties, we consider the action leading to the most important reward with a level of probability. An extension of UCB in the linear case is defined :

Input: k arms
for $t = 1, \dots T$ **do**
  $\hat{\theta}_t = A_t^{-1} b_t \quad \hat{\mu_t}(a) = \phi(a)^T \hat{\theta}_t$
  $B_t(a) = \hat{\mu_t}(a) + \sqrt{\phi(a)^T A_t^{-1} \phi(a)}$
  $a_t = \underset{a}{\operatorname{argmax}}\, B_t(a)$
end

3 Boltzmann Exploration

3.1 Non efficient approach

Boltzmann exploration, or exponential weighting is a common strategy in machine learning to balance exploration and exploitation. This approach is inspired by statistical physics and the notion of entropy. Actually, it's a mechanism used to add uncertainty. In the multi-armed bandit setting, we choose the arm k at time t with probability :

$$p_{t,k} = \frac{e^{\eta_t \hat{\mu}_{t,k}}}{e^{\sum_{j=1}^{K} \eta_t \hat{\mu}_{t,s}}}$$

We have to choose a schedule for the learning rate η_t which is now a hyper-parameter of the problem. Common schedules are the constant one $\eta_t = 1/C^2$, the logarithmic one $\eta_t = \log(t)/C^2$ and a square root increase $\eta_t = \sqrt{t}/C^2$. We notice that the learning rate has to increase in order to select the sub-optimal arms with less probability. The article [Ces+17] gives interesting result about this frequent heuristic. The following results use the empirical mean estimate :

$$\hat{\mu}_{t,k} = \frac{\sum_{s=1}^{t} r_{t,k} \mathbb{1}_{\{I_s = k\}}}{N_{t,k}}$$

In the case of the two-armed bandit, we can suppose that the arm n°1 is optimal and by denoting $\Delta = \mu_1 - \mu_2$, we have :

Proposition 1. *Let us assume that $\hat{\mu}_{t,k} = \mu_k$ for all t and k. If $\eta_t = o(\frac{log(t\Delta^2)}{\Delta})$, then the regret grows at least as fast as $R_T = \omega(\frac{log(T)}{\Delta})$*

Proposition 2. *There exists a 2-armed stochastic bandit problem with reward bounded in $[0;1]$, where Boltzmann exploration using any learning rate sequence η_t such that $\eta_t > 2log(t)$ for all $t \geq 1$ has regret $R_T = \Omega(T)$.*

The proposition 1 states that even if the estimation of the reward is perfect, the learning rate has to increase at least at a rate $o(\frac{\log(t\Delta^2)}{\Delta})$ else the regret is sub-optimal. The proposition 2 shows that if the learning rate increases faster than $2\log(t)$ then the regret grows linearly. This approach is not very efficient despite its wide use, the main drawback comes from the fact that the uncertainty is the same for arm and independent of the mean estimation quality.

3.2 Gumbel Trick

Instead of computing $p_{t,k}$ with the previous formula, we can use the property of the Gumbel distribution denoted $\mathcal{G}$. If $G \sim \mathcal{G}(\mu)$ then it's density function is $f(x;\mu) = exp(-(x-\mu) + \gamma + e^{-(x-\mu)+\gamma})$ and it's cumulative distribution function is $F(x;\mu) = \exp(-e^{-(x-\mu)+\gamma})$ with γ the Euler constant and μ it's mean.

Property : Let $x_1, \dots x_n \in \mathbb{R}$ and $G_1, \dots, G_n \sim \mathcal{G}(0)$ independent and uniformly distributed then sampling $i \in [\![1;n]\!]$ according to $p_i \propto exp(x_i)$ is equivalent to take :

$$i = \underset{j}{\text{argmax}} \{x_j + G_j\}$$

Demonstration : We have $x_j + G_j \sim \mathcal{G}(x_j)$, we denote z_j the realization of $x_j + G_j$ and we compute the probability that the realization associated to the arm i is the greatest.

$$\mathbb{P}(\{z_i > z_j \ \forall \ j \neq i\}) = \prod_{k \neq i} \exp(-e^{-(z_k - x_k)+\gamma})$$

Then we can integrate over $z_i \in \mathbb{R}$ in order to compute the probability that the arm i is chosen and we denote $\mathcal{I}$ this event :

$$\begin{aligned}
\mathbb{P}(\mathcal{I}) &= \int_{\mathbb{R}} \exp(-(z_i - x_i) + \gamma + e^{-(z_i - x_i)+\gamma}) \prod_{k \neq i} \exp(-e^{-(z_i - x_k)+\gamma}) \mathrm{d}z_i \\
&= \int_{\mathbb{R}} \exp(-(z_i - x_i) + \gamma + e^{-z_i+\gamma} \sum_k e^{x_k}) \mathrm{d}z_i \\
&= \frac{\exp(x_i)}{\sum_k \exp(x_k)}
\end{aligned}$$

Since $\int \exp(-x - ae^{-x})\mathrm{d}x = \frac{\exp(-ae^{-x})}{a}$ and we find the common expression of the softmax distribution.

Boltzmann exploration can be easily reformulated, the action chosen at time t is :

$$a_t = \underset{k}{\text{argmax}} \{\eta_t \hat{\mu}_{t,k} + G_{t,k}\}$$

It means that we draw random perturbation $G_{t,k}$ according to Gumbel distribution at each time step and for each arm.

3.3 Right exploration

As we have seen previously, the uncertainty has the same scale for each arm which leads to sub-optimal results. The idea now is to scale each perturbation by a different factor characterizing the quality of the mean estimator. We define $\beta_{t,k} = \sqrt{C^2/N_{t,k}}$ and we choose the arm at time t according to the following rule :

$$a_t = \underset{k}{\text{argmax}} \{\hat{\mu}_{t,k} + \beta_{t,k} G_{t,k}\}$$

This new exploration scheme is called Boltzmann-Gumbel exploration and it is still straightforward to implement. However with this formulation we don't have access to a closed form formula for the probabilities $p_{t,k}$. [Ces+17] provide the following results :

Proposition 3. *Assume the rewards of each arms are σ^2-subgaussian. Then, the regret of Boltzmann-Gumbel exploration with $C = \sigma$ satisfies $R_T \leq 200\sigma\sqrt{KT}logK$*

Proposition 4. *For any K and T such that $\sqrt{K/T}logK \leq 1$, there exists a bandit problem with rewards bounded in $[0; 1]$ where the regret of Boltzmann-Gumbel exploration with $C = 1$ is at least $R_T \geq \frac{1}{2}\sqrt{KT}logK$*

The bound provided by this approach are better in time than the bound provided by UCB which is $\sqrt{KT\log T}$ but sub-optimal in the number of arm. However, in most application, $T \geq K$ we obtain a better bound with Boltzmann-Gumbel exploration. Besides, the bound provided by proposition 3 is tights since proposition 4 shows that there exists a problem with a regret similar to the bound up to a scaling factor.

We can illustrate this on a simple example, we consider a two-armed bandits, the reward distribution of each arm follow a Bernoulli distribution of parameter μ_1 and μ_2 with $\mu_1 = 1/2 + \Delta$, $\mu_2 = 1/2$ and $\Delta = 0.01$. The horizon of the experiment is $T = 10^6$. We present in 2 the average regret over 10 runs of each methods : Boltzmann-Gumbel exploration, UCB, and the three schedule proposed for softmax exploration. We notice that BGE algorithm gives the best regret for $C = 0.25$, which is coherent since the rewards follow a Bernoulli distribution which is 1/4-subgaussian. This method clearly outperforms the other, we also notice that UCB and BE-sqrt performs quite well for $C \approx 1$. The 3 is an unfavorable situation, the first 5000 draws of the optimal arm are 0, we notice that the algorithm still choose the optimal arm and the regret is in the same order as the previous one. However, in this situation the classic softmax exploration with a learning rate proportional to $\sqrt{t}$ does not detect the optimal arm and choose at each run the sub-optimal one. This second experiments demonstrate both the robustness and the performances of Boltzmann-Gumbel exploration.

Figure 1: Average regret over 10 runs in function of C

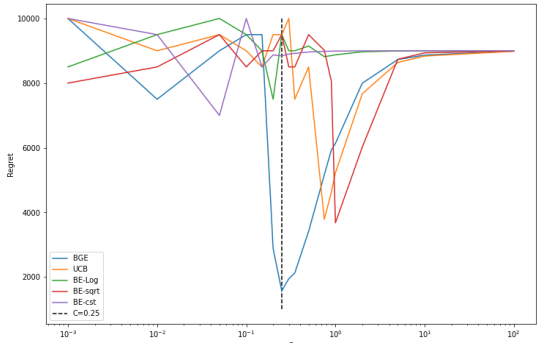

Figure 2: Normal situation

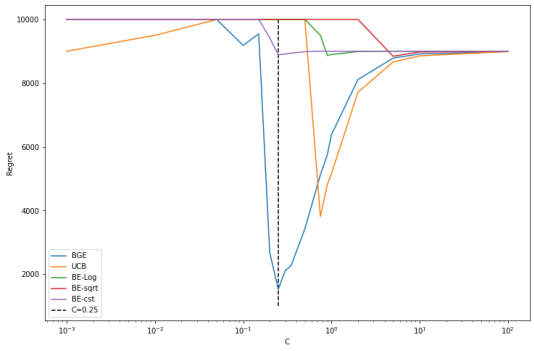

Figure 3: Unfavorable situation

4 Gradient Descent

As we have seen in LinUCB, the vector of weights θ_t is the minimizer of a convex function f_t. It can be solve analytically but it requires matrix inversion therefore we use gradient descent. In our case, the vector θ belongs to $\mathbb{R}^K$ so it's an unconstrained minimization problem.

$$\theta_t = \underset{\theta}{\operatorname{argmin}} \sum_{s=1}^{t} (r_s - \phi(a_s)^T\theta)^2 + \lambda\|\theta\|_2^2$$

So we define the loss function $l_t(\theta) = (r_t - \phi(a_t)^T\theta)^2 + \lambda\|\theta\|_2^2$ and its gradient : $\nabla_\theta l_t(\theta) = 2(r_t - \phi(a_t)^T\theta) \times \phi(a_t) + 2\lambda\theta$. It's the loss associated to the sample obtain at time t, we define consequently the loss $l_{t_1,t_2}(\theta) = \sum_{s=t_1+1}^{t_2} l_s(\theta)$, the cumulative loss between the time step t_1 and t_2. Since we view only noisy observation of the reward, we have access to a noisy gradient $\widetilde{\nabla}_\theta l_t(\theta)$ verifying $\mathbb{E}(\widetilde{\nabla}_\theta l_t(\theta)) = \nabla l_t(\theta)$ [Haz19] proposed the online stochastic gradient descent which solve the minimization problem with θ belonging to a convex set $\mathcal{C}$.

Input: step size η_t, $\theta \in \mathcal{C}$
for $t = 1, \dots T$ **do**
  $\widetilde{\nabla}_t = \nabla_\theta l_t(\theta_t)$
  $\overset{o}{\theta}_{t+1} = \theta_t - \eta_t \widetilde{\nabla}_t$
  $\theta_{t+1} = \Pi_{\mathcal{C}}(\overset{o}{\theta}_{t+1})$
end
return $\bar{\theta} = \frac{1}{T}\sum_{s=1}^{T} \theta_s$

The paper [RSS12] states that this algorithm converges for strongly convex functions.

Proposition 5. *Assume a function f λ-strongly convex and α-smooth defined on a convex set, with $\mathbb{E}(\|\hat{\mu}_t\|_2^2) \leq G$. For an appropriated learning rate $\eta_t = \frac{1}{\lambda t}$, then*

$$\mathbb{E}(\|\bar{\theta}_T - \theta^\star\|_2^2) \leq \frac{4G^2}{\lambda^2 T}$$

Since our loss function is defined with a regularization coefficient λ it is λ-strongly convex therefore we are sure to have a convergence rate in $\mathcal{O}(\frac{1}{\sqrt{T}})$ since the proposition above holds with the squared norm.

5 Combining BGE and SGD

We propose to combine both stochastic gradient descent and Boltzmann-Gumbel exploration. As we saw previously, estimating θ can be achieved with linear regression but it's costly therefore we use gradient descent to do the estimation and reduce the algorithm complexity. Unfortunately, with this approach we don't have access to easy confidence bound enabling us to deduce the bonus term for each arm. However, Boltzmann-Gumbel exploration may be used regardless of the estimation method used. We combine both method to provide the algorithm described below. We choose to do a batch gradient descent with batch size equal to τ, which is an hyper-parameter that we can fine-tuned. Doing a batch gradient descent permits a better estimation of the gradient which is noisy. The first τ actions are chosen totally randomly then we estimate $\hat{\theta}_1$ with the τ first reward by doing a linear regression. With this definition of $\hat{\theta}_1$, the estimation should be not too far from the optimal value. We hope that the convergence is better with a good initialization which provides a more robust algorithm. Then we choose the arm as described previously with BGE exploration using $\bar{\theta}$ which is the average of the $\hat{\theta}$ sequence. The learning rate is

defined by $\eta_i = \eta/i$, and decreases after each gradient descent.

Input: step size η_t, Batch size τ, hyper-parameter of exploration C
$N = (0, \dots, 0) \in \mathbb{R}^K$
for $t = 1, \dots \tau$ **do**
  Choose $a_t \in [\![1; K]\!]$ randomly
  observe r_t
  $N[a_t] = N[a_t] + 1$
end
Estimate $\hat{\theta}_1 = A_t^{-1} b_t$
$\bar{\theta}_1 = \hat{\theta}_1$
for $t = \tau + 1, \dots T$ **do**
  $i = \lfloor t/\tau \rfloor$
  $\hat{\mu}_{t,k} = \phi(k)^T \bar{\theta}_i$ **for** $k \in [\![1; K]\!]$
  $\beta_{t,k} = \sqrt{C^2/N[k]}$ **for** $k \in [\![1; K]\!]$
  $a_t = \underset{k}{\text{argmax}}\{\hat{\mu}_{t,k} + \beta_{t,k} G_{t,k}\}$ avec $G_{t,k} \sim \mathcal{G}(0)$
  draw arm a_t and observe r_t
  if $t\%\tau = 0$ **then**
    $\eta_i = \eta/i$ Compute $\nabla_\theta l_{t-\tau,t}$ and update $\hat{\theta}_i \leftarrow \hat{\theta}_{i-1} - \eta_i \nabla_\theta l_{t-\tau,t}$
    $\bar{\theta}_i = 1/i \sum_{l=1}^{i} \hat{\theta}_l$
  end
end

6 Another baseline algorithm : KL-UCB

In [GC11] paper, they adapt the traditional multi-armed bandit algorithm : KL-UCB for bounded rewards, to a more general family of rewards called exponential distributions. This is a parameterized distribution where the associated probability density function is :

$$p_\theta(x) = \exp\{x\theta - b(\theta) + c(x)\}$$

In our linear setting, the rewards are drawn from a linear model with some Gaussian noise, so the rewards are actually drawn from a Gaussian distribution. Gaussian distribution belong to the exponential family.

$$p_\theta(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\{-\frac{(\mu - x)^2}{2\sigma^2}\}$$

Thus we can identify :

$$\begin{cases} \theta = \frac{\mu}{\sigma^2} \\ b(\theta) = \frac{1}{2}\theta^2\sigma^2 = \frac{\mu^2}{2\sigma^2} \\ c(x) = \frac{x^2}{2\sigma^2} \end{cases}$$

As described in the paper [GC11], we can adapt the usual KL-UCB by switching $d(\mu(\beta), \mu(\theta))$ with $\mu(\beta)(\mu(\beta) - \mu(\theta)) - b(\theta) + b(\beta)$.

Moreover, the arm pulled at each step is :

$$I_t = \text{argmax}_k \max \left\{ \mu(\theta) : d(\hat{\mu}_{k,t}, \mu(\theta)) \leq \frac{\log(1 + t\log^2(t))}{N_{k,t}} \right\}$$

If the exponential model is Gaussian (with known sigma) we have :

$$d(\hat{\mu}_{i,t}, \mu(\theta)) = \hat{\mu}_{i,t}(\frac{\hat{\mu}_{i,t}}{\sigma^2} - \frac{\mu(\theta)}{\sigma^2}) - \frac{\hat{\mu}_{i,t}^2}{2\sigma^2} + \frac{\mu(\theta)^2}{2\sigma^2} = \frac{(\hat{\mu}_{i,t} - \mu(\theta))^2}{2\sigma^2}$$

Thus we need to compute :

$$I_t = \text{argmax}_k \max \left\{ \mu(\theta) : \frac{(\hat{\mu}_{i,t} - \mu(\theta))^2}{2\sigma^2} \leq \frac{\log(1 + t\log^2(t))}{N_{k,t}} \right\}$$

Let's compute

$$\max \left\{ \mu(\theta) : \frac{(\hat{\mu}_{k,t} - \mu(\theta))^2}{2\sigma^2} \leq \frac{\log(1 + t\log^2(t))}{N_{k,t}} \right\}$$

.

The function $\mu \mapsto \frac{(\hat{\mu}_{k,t}-\mu)^2}{2\sigma^2} - \frac{\log(1+t\log^2(t))}{N_{k,t}}$ is convex and the objective function of our problem is linear. Thus at the optimum, we have :

$$\frac{(\hat{\mu}_{k,t} - \mu^\star)^2}{2\sigma^2} - \frac{\log(1 + t\log^2(t))}{N_{k,t}} = 0$$

This equation has two solution and we take the maximum one so we get :

$$\mu^\star_{k,t} = \max \left\{ \hat{\mu}_{k,t} + \sqrt{2\sigma^2 \frac{\log(1 + t\log^2(t))}{N_{k,t}}}, \hat{\mu}_{k,t} - \sqrt{2\sigma^2 \frac{\log(1 + t\log^2(t))}{N_{k,t}}} \right\} = \hat{\mu}_{k,t} + \sqrt{2\sigma^2 \frac{\log(1 + t\log^2(t))}{N_{k,t}}}$$

One another hand, we have still not used that the rewards come from a linear model (except for the Gaussian assumption).

We thus estimate $\hat{\mu}_{i,t}$ using a least square regression as we did for the BGE algorithm.

$$\begin{cases} \hat{\theta}_t = \underset{\theta}{\text{argmin}} \ \sum_{s=1}^{t} (r_s - \phi(a_s)^T\theta)^2 + \lambda\|\theta\|_2^2 \\ \hat{\mu}_{k,t} = \phi(k)^T\hat{\theta}_t \end{cases}$$

We can now write the algorithm adapted from the KL-UCB-Exponential one. Only the choice of the new action changes.

Input: step size η_t, Batch size τ, σ standard deviation
$N = (0, \dots, 0) \in \mathbb{R}^K$
for $t = 1, \dots \tau$ **do**
  Choose $a_t \in [\![1; K]\!]$ randomly
  observe r_t
  $N[a_t] = N[a_t] + 1$
end
Estimate $\hat{\theta}_1 = A_t^{-1} b_t$
$\bar{\theta}_1 = \hat{\theta}_1$
for $t = \tau + 1, \dots T$ **do**
  $i = \lfloor t/\tau \rfloor$
  $\hat{\mu}_{t,k} = \phi(k)^T\bar{\theta}_i$ **for** $k \in [\![1; K]\!]$
  $b_{t,k} = \sqrt{2\sigma^2 \frac{\log(1+t\log^2(t))}{N_{k,t}}}$ **for** $k \in [\![1; K]\!]$
  $\mu^\star_{t,k} = \hat{\mu}_{t,k} + b_{t,k}$ **for** $k \in [\![1; K]\!]$
  $a_t = \underset{k}{\text{argmax}}\{\mu^\star_{t,k}\}$
  draw arm a_t and observe r_t
  if $t\%\tau = 0$ **then**
    $\eta_i = \eta/i$ Compute $\nabla_\theta l_{t-\tau,t}$ and update $\hat{\theta}_i \leftarrow \hat{\theta}_{i-1} - \eta_i \nabla_\theta l_{t-\tau,t}$
    $\bar{\theta}_i = 1/i \sum_{l=1}^{i} \hat{\theta}_l$
  end
end

7 The last baseline algorithm : TS-SGD

The last algorithm we used to compare the results is the Thompson Sampling adapted for generalized linear models [DHS20]. In the paper, they also use a stochastic gradient descent to estimate the maximum likelihood estimator associated with the generalized linear model. Contrary to us, they don't have a ridge parameter λ to ensure the proper convergence, but they keep a convex set $\mathcal{C} = \{\theta : \|\theta - \hat{\theta}_\tau\| \leq 2\}$. Moreover, in order to sample the reward, they draw a parameter $\theta_j^{TS} \sim \mathcal{N}(\bar{\theta_j}, A_j)$, where $\bar{\theta_j}$ is the average of the previous j estimations $\tilde{\theta_j}$. A_j is a matrix controlling the exploration, since $A_j \approx \sum_{s=1}^{T} \phi(a_s)\phi(a_s)^T$, is a covariance matrix. They demonstrated a time complexity of $\mathcal{O}(Td)$ and a theoretical regret of $\mathcal{O}(\sqrt{Td})$. These time complexity and regret ensure the algorithm to be state of the art for generalized linear model bandits. Thus we will compare their algorithm to ours empirically.

8 Experimental results

In this part, we illustrate the effectiveness of the proposed method and compared all the algorithm on a very simple problem. We set $K = 10$ and defined by hand the features of the 10 arms which have three dimensions. We defined the true value of the vector θ and by pulling arm k, the reward observed is a realization of $\phi(k)^T\theta_{True} + \epsilon$ with $\epsilon \sim \mathcal{N}(0, 0.5)$. This setting allow us to check the convergence of the sequence θ_t and to compute the average regret of the algorithm easily. We ran a grid search over the hyper-parameter of each model, $\eta \in (1\times10^{-4}, 1\times10^{-3}, 1\times10^{-2}, 1\times10^{-1}, 5\times10^{-1}, 1)$, $\tau \in (10, 25, 50, 100)$ and $C \in (1 \times 10^{-1}, 0.25, 0.5, 1, 5)$. In the graph below, we plot the average cumulative regret over 10 runs and plot the trajectory for the best hyper-parameter. At the beginning of the plot, we see that the regret grows linearly since we choose the action randomly in order to estimate correctly θ. We notice that Linear UCB always choose the right arm after the first random actions therefore the regret is globally constant (sometime the wrong arm is pulled). For BGE exploration, the regret is better at the beginning but still grows, it means that the exploration is still acceptable therefore the agent pulls quite often a wrong arm but it achieves a good regret compared to the other methods.

Figure 4: Average cumulative regret over 10 runs

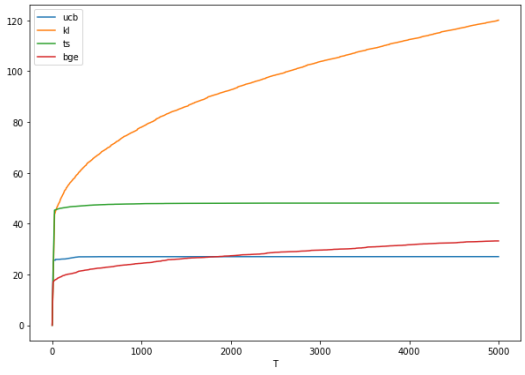

Figure 5: Average cumulative regret over 10 runs

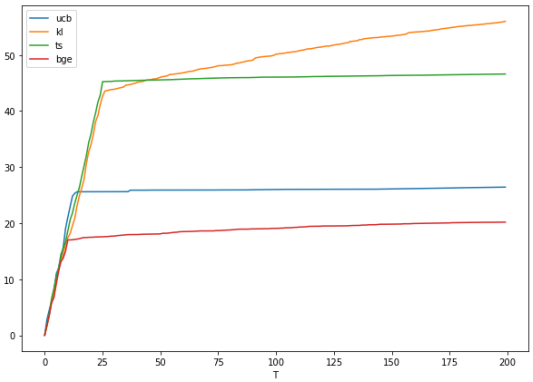

Figure 6: Average cumulative regret over 10 zoomed

We can ensure that the algorithm converges, we plot in 8 the l2 distance between the theta estimated and the ground truth, we do not observe the convergence towards 0 because of the regularization coefficient λ set to 0.1. Finally in 9, we observe the repartition of the arm pulled at the end of the trajectory and we notice that the arm n°6 is the one that is quite often chosen (the arm n°5 is the optimal one) and it explains the constant growth of the regret of BGE algorithm.

Figure 7: Boltzmann-Gumbel exploration

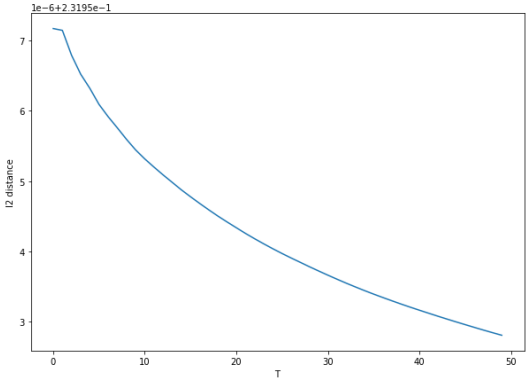

Figure 8: convergence of RMSE between $\bar{\theta}_t$ and $\theta^\star$

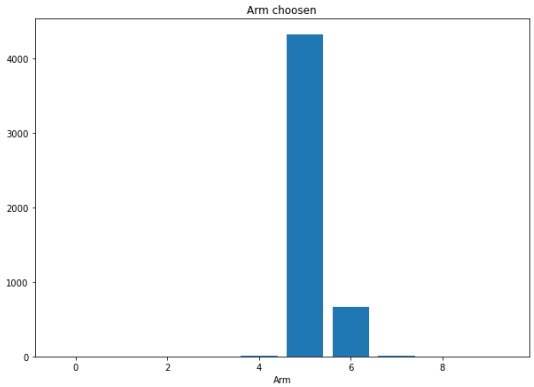

Figure 9: Final arm selection for a trajectory

We did hyper-parameter search but we can analyze the influence of each hyper-parameter. The figure 12 shows that when η is too large then it leads to an exploding regret but when η is below 10^{-1} there are no differences. When it comes to the parameters controlling the exploration C, we see that BGE can perform badly when the parameter is too high therefore we select too many sub-optimal arms. However, LinUCB is less sensitive too this parameters and we don't notice any difference.

Figure 10: Hyper-parameter sensitivity

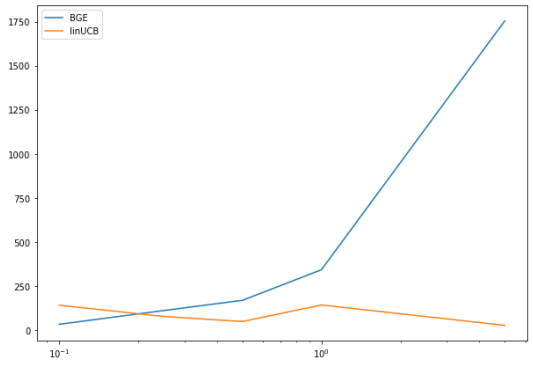

Figure 11: C sensitivity

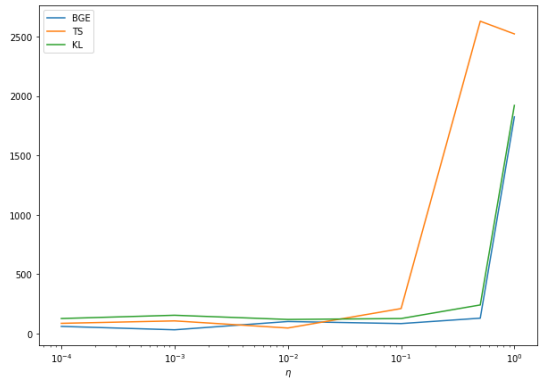

Figure 12: η sensitivity

In order to illustrate the goal of the project (avoiding the complexity of LinUCB) we can plot the average computation time for BGE and LinUCB for several horizons. The computation time for LinUCB is quadratic in time while the BGE exploration leads to a linear complexity (in time) so in most real world problem it's useful since the number of arm may be very important.

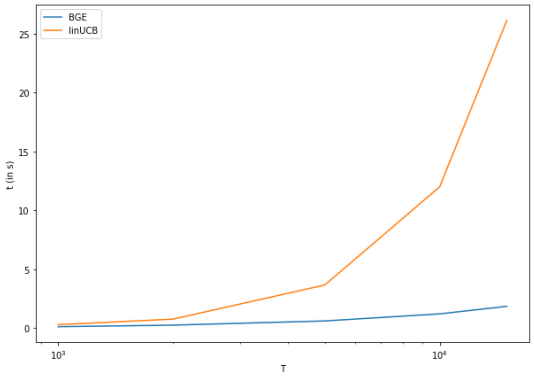

Figure 13: Average computation times (in s) in function of T

9 Contextual Bandits and MovieLens Dataset

Our algorithm and the benchmark algorithms can easily be adapted to work in the setting of contextual linear bandits. In this setting, if we choose the arm a for a user u at time t, we suppose that the reward has the following form :

$$r_t(a,u) = \phi(a,u)^T\theta^\star + \epsilon_t \quad \text{with} \quad \mathbb{E}(\epsilon_t) = 0$$

This is equivalent to say, that we construct a bandit algorithm for each user u but the parameter $\theta^\star$ is shared across all users (instead of depending of each user θ_u).

In this setting, we try to estimate $\theta^\star$ at each step t as :

$$\theta_t = \underset{\theta}{\operatorname{argmin}} \sum_{s=1}^{t} (r_s - \phi(a_s,u_s)^T\theta)^2 + \lambda\|\theta\|_2^2$$

Adapting our algorithm is straightforward (using $\phi(a,u)$ instead of $\phi(a)$). To test it on a real dataset, we used the MovieLens-100k dataset and we tried to create a basic recommender system. For each encountered user (and their contextual information), the policy will recommend the best movie (which are treated as arms), and observe the reward that will be used for online learning in order to recommend better movies. We have a dataset of more than 1600 distinct movies and 940 distinct users, the problem lies in the fact that the users have not seen all the the movies therefore the matrix with the notation is very sparse. We filter the matrix to conserve only the 30 movies the most seen and to the 200 users that have rated the most movies so the matrix is less sparse. When we provide a a recommendation to the user, if we recommend a movie that has not been ranked by the user we do not observe a loss. So this approach is not perfect but provide only loss when the information is known.

Moreover, to estimate the convergence of the $\theta^\star$ parameter, we do a linear regression on the full dataset to define θ_{lin} which is the best linear approximation using least squares. We then plot $\|\theta_{lin} - \theta_j\|_2$ at each step j we should give a good estimation of $\|\theta^\star - \theta_j\|_2$

- We define the reward using users' ratings. The reward is defined to be 1 if the user's rating for a movie is 5, and 0 otherwise (4 or below).
- We used the top 30 movies in terms of number of user ratings.

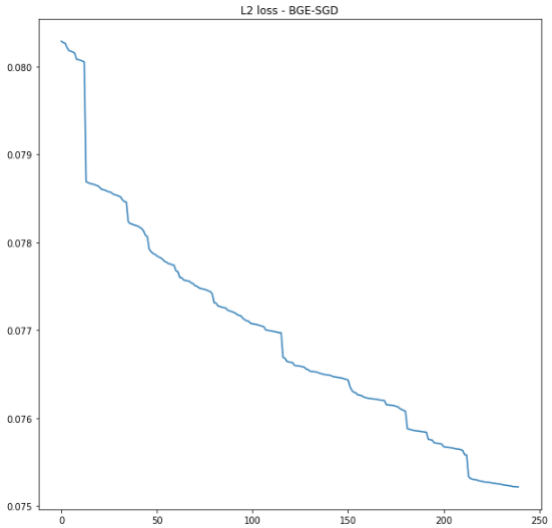

Figure 14: L2 loss between the estimated θ and the "optimal" θ

In the figure 14 we notice that with our estimation $\bar{\theta}_t$ seems to converge toward the proxy that we have defined above. We do not plot the regret since we only compute the loss when the user has already provided a rating while in a real application, we could know if the user has watched the movie after proposing it.

10 Conclusion

We try to provide an algorithm with an affordable computational complexity by combining online stochastic gradient descent and Boltzmann-Gumbel exploration. We analyze empirically this algorithm on a simple problem and compare it with linUCB, KL-UCB (with SGD) and SGD-TS. We studied on this problem the effect of hyper-parameters controlling the exploration, the gradient descent and the batch size. Then we used our algorithm that we extended to contextual bandits on a real dataset for a recommender system. With more time, we would like to go deeper in the mathematical aspect of the problem to see if we can provide an upper-bound of the regret.

References

[GC11] Aurélien Garivier and Olivier Cappé. "The KL-UCB algorithm for bounded stochastic bandits and beyond". In: *Proceedings of the 24th annual conference on learning theory*. JMLR Workshop and Conference Proceedings. 2011, pp. 359–376.

[RSS12] Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. *Making Gradient Descent Optimal for Strongly Convex Stochastic Optimization*. 2012. arXiv: 1109.5647 [cs.LG].

[Ces+17] Nicolò Cesa-Bianchi et al. *Boltzmann Exploration Done Right*. 2017. arXiv: 1705.10257 [cs.LG].

[Haz19] Elad Hazan. *Introduction to Online Convex Optimization*. 2019. arXiv: 1909.05207 [cs.LG].

[DHS20] Qin Ding, Cho-Jui Hsieh, and James Sharpnack. *An Efficient Algorithm For Generalized Linear Bandit: Online Stochastic Gradient Descent and Thompson Sampling*. 2020. arXiv: 2006.04012 [cs.LG].

Appendix

Experimental settings

We defined $\phi(a)$ for all actions (arms) and defined $\theta^\star = (-1, -1, 3)^T$ and the true mean of the reward for each arm is the following vector : $(-6, -4.08, -2.56, -1.44, -0.71, -0.38, -0.44, -0.90, -1.75, -3)^T$. The optimal arm is the n°5 and the closest one is the n°6 with $\Delta_6 = 0.06$.

Mirror gradient descent

Since the goal of the project is to avoid matrix inversion, we can estimate θ with gradient descent. If the set Θ has a specific structure like the simplex for example, we can regularize the gradient to take into account this structure. It leads to mirror descent.

Definition 1. *Let $F : \Omega \to \mathbb{R}$ a convex and differentiable function defined on a closed convex set then the Bregman divergence with respect to F is defined by :*

$$D_F(x||y) = F(x) - F(y) - < \nabla F(y), x - y >, \quad (x, y) \in \Theta \times \Theta$$

The Bregman divergence is the difference between the regularization function evaluated in x and the first order approximation of this function. If the regularization function is norm 2, $F(x) = \|x\|_2^2$ then the Bregman divergence is the distance function $D_F(x||y) = \|x - y\|_2^2$. If the regularization function is the negative entropy $F(x) = \sum_i x(i)\log(x(i))$ the Bregman divergence is then the generalized Kullback-Leibler divergence : $D_F(x||y) = \sum_i x(i)\log(\frac{x(i)}{y(i)}) - \sum_i x(i) + \sum_i y(i)$. The updating rules is the following :

$$x_{k+1} = \underset{x \in \mathcal{C}}{\operatorname{argmin}} \{< x_k, \nabla F(x_k) > + \eta_k D_F(x||x_k)\}$$