Bandit Optimization Algorithms

1st Lakshay Tyagi
170360

2nd Prakhar Srivastava
170486

Mode
Application

***Abstract*—The Online Learning problem can be formulated as a repeated game between an Online Player and Adversary in which at each time step an online player takes an action and suffers a loss returned by the Adversary. The Player should play in such a way that the total loss is minimized. Bandit algorithms are a class of Online optimization problems where the feedback received is limited and is received as a loss at the queried data point.**

In our term paper we will be implementing four Bandit Convex Optimization Algorithms while comparing their performance. We present a novel analysis on the effect on regret bound by various parameters. And also experimentally verify the theoretical bound given by the authors.

***Index Terms*—$\mathbf{x_t}$ = Vector in convex set S, $\mathbf{f_t}$ = Convex function at time-step t, Bandit Convex Optimization(BCO), SCRIBLE(Self Concordant Regularization for Bandit Learning), Online Gradient Descent(OGD)**

I. Introduction

Online Learning problems are a class of Machine Learning problems in which data becomes available in a sequential manner and an agent or player has to be trained at each step to predict the data in the future. Bandit Problems are a subclass of Online Learning problems where the feedback received from the Oracle or Adversary is limited. Bandit Convex Optimization can be seen as a repeated game which can be described in the following framework : At each time step t, the Player chooses $\mathbf{x_t} \in \mathcal{K}$, After, this choice the adversary chooses a convex cost function $\mathbf{f_t} \in \mathcal{F}$. The cost or penalty for this chosen cost function $\mathbf{f_t}(\mathbf{x_t})$ is then revealed to the player. The player's choices should be such that the regret is minimized. The difference between Bandit Convex Optimization (BCO) Problems and simple Online Convex Optimization (OCO) Problems is that compared to OCO where the player has access to gradient for $\mathbf{f_t}$ over $\mathcal{F}$, in BCO the player only has point feedback $\mathbf{f_t}(\mathbf{x_t})$ at each time step t. The objective of each Bandit Optimization Algorithm is to minimize the total Regret over all time-steps. The regret can be thought of as the regret of the player over all time-steps which is the difference of cost between the taken action and the optimal action. The regret for an Algorithm $\mathcal{A}$ can be written as

$$regret(\mathcal{A}) = \sup_{\{\mathbf{f_1},\ldots,\mathbf{f_T}\}\subseteq\mathcal{F}} \left[\sum_{t=1}^{T} \mathbf{f_t}(\mathbf{x_t}) - \min_{\mathbf{x}\in\mathcal{K}} \sum_{t=1}^{T} \mathbf{f}(\mathbf{x}) \right]$$

Identify applicable funding agency here. If none, delete this.

Where T is the total number of timesteps. In this work we restrict our domain to linear cost functions which is a variation of BCO called Linear Bandit Convex Optimization. We will be exploring 4 such BCO Algorithms and comparing their results.

II. Algorithms:

We have implemented four Algorithms for the Bandit Optimization Problem. The first two algorithms are general in nature and can be applied for any convex functions while the last two are only for linear optimization. Algorithms are described below:

- Bandit Online Gradient Descent [1]
- Bandit Online Gradient Descent wihout a Gradient [2]
- SCRIBLE (Self Concordant Regularization in Bandit Learning) [3]
- Bandit Online Linear Optimization [4]

Algorithm 1: Bandit Online Gradient Descent[1]

Parameters: $\eta, \delta > 0$ and Convex Set $S \subset \mathbb{R}^d$;
Initialization: $\theta_\mathbf{1} = 0$;
for $t = 1, 2 \ldots T$ **do**
  let $\mathbf{w_t} = \text{argmin}_{\mathbf{w}\in S} \|\mathbf{w} - \eta\theta_\mathbf{t}\|_2$;
  pick $\mathbf{v_t} \sim U_{sp}$;
  predict $\mathbf{w_t} + \delta\mathbf{v_t}$ and receive $\mathbf{f_t}(\mathbf{w_t} + \delta\mathbf{v_t})$;
  set $\mathbf{z_t} = \frac{1}{\delta}\mathbf{f_t}(\mathbf{w_t} + \delta\mathbf{v_t})\mathbf{v_t}$;
  update $\theta_{\mathbf{t+1}} = \theta_\mathbf{t} - \mathbf{z_t}$;
end

Algorithm 2: Bandit Online Gradient Descent without a Gradient[2]

$w_0 = \mathbf{0}$;
Parameters: $\delta, \mu > 0$ and $0 < \alpha < 1$ Convex Set $S \subset \mathbb{R}^d$;
for $t = 1, 2 \ldots T$ **do**
  u_t = Uniform and random unit vector;
  set $x_t = w_{t-1} + \delta * u_t$;
  get cost $c_t = f_t(x_t)$ where f_t is a convex function;
  $w_{t+1} = w_t - \nu * c_t * u_t$;
  set $w_{t+1} = \mathbf{P}_{(1-\alpha)S}(w_t)$
end

Algorithm 3: SCRIBLE[3]

Parameters: $\eta > 0$, Convex Set $\mathcal{K} \subset \mathbb{R}^d$ and Self Concordant Barrier R ;

Initialization: Initialize $\mathbf{x_1}$ such that $\nabla \mathcal{R}(\mathbf{x_1}) = 0$;

for $t = 1, 2 \dots T$ **do**

Let $\mathbf{A_t} = \left[\nabla^2 \mathcal{R}(\mathbf{x_t})\right]^{-\frac{1}{2}}$;

Pick $\mathbf{u_t} \in \mathbb{S}$ uniformly, and set $\mathbf{y_t} = \mathbf{x_t} + \mathbf{A_t u_t}$;

Play $\mathbf{y_t}$, observe and suffer loss $\mathbf{f_t}(\mathbf{y_t})$;

Let $\mathbf{g_t} = \mathbf{f_t}(\mathbf{y_t}) {\mathbf{A_t}}^{-1} \mathbf{u_t}$;

Update

$$\mathbf{x_{t+1}} = \underset{\mathbf{x} \in \mathcal{K}}{\operatorname{argmin}} \left[\eta \sum_{t=1}^{T} {\mathbf{g_t}}^T \mathbf{x} + R(\mathbf{x}) \right]$$

;

end

Algorithm 4: Bandit Online Linear Optimization[4]

Parameters: $\eta > 0$ and $\theta-$self-concordant $R(x)$;

$w_0 = \operatorname{argmin}_{\mathbf{w} \in S}[R(w)]$;

for $t = 1, 2 \dots T$ **do**

$\mathbf{E}, \lambda =$ Eigenvectors and Eigenvalues of $\nabla^2 R(w_t)$;

$\mathbf{i_t} =$ Random column vector of $\mathbf{E}$, and $\epsilon = \pm 1$ with probability 0.5;

$\mathbf{y_t} = \mathbf{w_t} + \epsilon_t \lambda_{i_t}^{-\frac{1}{2}} \mathbf{e}_{i_t}$;

$\hat{\mathbf{f}}_\mathbf{t} = d * (\mathbf{f_t^T w_t}) \epsilon_t \lambda_{i_t}^{\frac{1}{2}}$;

set $\mathbf{w_{t+1}} = \operatorname{argmin}_{\mathbf{w} \in S} \left[\eta \sum_{s=1}^{t} \hat{\mathbf{f}}_\mathbf{s}^{\ T} \mathbf{x} + R(\mathbf{w}) \right]$

end

III. EXPERIMENTAL SETUP:

A. Algorithm 1:

The convex set S which is the set of actions available to the player is chosen as a Norm Ball which is the set $\mathbf{w} = \{\mathbf{w} \in \mathbb{R} : \|\mathbf{w}\| <= 1\}$. The total number of time steps for all experiments is chosen to 1000. The Adversary is provided with a finite set of possible cost functions and at each time-step it chooses a random cost function from this set of cost functions and returns the calculated cost to the Player. U_{sp} from which we sample $\mathbf{v_t}$ is the Unit Sphere. For this we sample from a d-dimensional normal distribution and normalize the sampled values. Since, the Gaussian is a symmetrical distribution this is equivalent to sampling from the unit sphere. The parameters η and δ are varied and the sensitivity of the algorithm to these parameters is measured. Finally, performance of the algorithm to theoretical bounds given in [1] is compared.

B. Algorithm 2:

Set S contains a ball of radius 'r' and is contained in ball of radius 'R'. $\mathbf{w} \in S$ and $w \in [-C, C]$ and $S \subset \mathbb{R}^d$.
The optimal parameters are: $\mu = \frac{R}{C\sqrt{T}}$, $\delta = (\frac{rR^2d^2}{12T})^{1/3}$ and $\alpha = (\frac{3Rd}{2r\sqrt{T}})^{1/3}$.[2]

For r = 1, R = 10, n = 2000, C = 4 and d = 5 we get $\alpha = 0.9$, $\delta = 0.27$ and $\nu = 0.024$. The player chooses the vector according to the algorithm. After the vector is chosen, to simulate the online environment a random linear function is revealed which is used to evaluate the cost. We set the dimension of the vector space and the number of time-steps. The optimal parameters are calculated using these. We then vary this parameters to see how these parameters affect our regret and to verify the theoretical bound given in [2].

C. Algorithm 3:

The set K which is the set of feasible actions is chosen as a Norm Ball which is the set $\mathbf{w} = \{\mathbf{w} \in \mathbb{R} : \|\mathbf{w}\| <= 1\}$. The function $\mathcal{R}$ which is the Concordant Barrier Regularization function is chosen as $\mathcal{R} = -\sum_{i=1}^{d} \log(x_i)$ which is a function from $\mathbb{R}^d \rightarrow \mathbb{R}$. For this function the Hessian $\nabla^2 \mathcal{R}$ is a diagonal matrix with diagonal elements $\nabla^2 \mathcal{R}_{i,i} = (\frac{1}{x_i^2})$. The total number of time steps for all experiments is varied from 1000-2000 depending on the application. The set $\mathbb{S}$ is the unit sphere to sample from which we use the same method as used for Algorithm 1. The Adversary is provided with a finite set of possible cost functions and at each time-step it chooses a random cost function from this set of cost functions and returns the calculated cost to the Player. The parameters η is varied and the sensitivity of the algorithm to these parameters is measured. Finally, performance of the algorithm to theoretical bounds given in [3] is compared.

D. Algorithm 4:

The setup for Algorithm 4 is same as 3. The difference between the two algorithms is that the Algorithm 3 uses the inverse of the Hessian of the Regularizer while Algorithm 4 uses eigen-decomposition of the Regularizer. We have to start with the optimal point as needed for the interior point method. We then perform the eigen decomposition on the hessian of the regularizer. A eigen vector is then randomly chosen to specify in which direction we have to update our chooses vector. Adversary then reveals a random cost function which the player suffers. The sensitivity of the algorithm is then measured to the parameters and also we verify the theoretical bound.

IV. SIMULATION AND NUMERICAL RESULTS:

A. Bandit Online Gradient Descent:

1) Parameter Sensitivity Analysis:

a) Variation with η : Bandit OGD Algorithm is not super sensitive to variation in η as long as we take some intermediate value. Very low values or very high values give bad performance. A modest value of η from 0.5-2 gives decent performance. The variations with η are shown in Fig 1

b) Variation with δ: Bandit OGD is also not very sensitive to the value of δ chosen as long as we choose an intermediate value. Values of δ from 0.1 to 1 give good performance with very large or very small values giving poor performance. The variation with δ is shown in Fig 2.

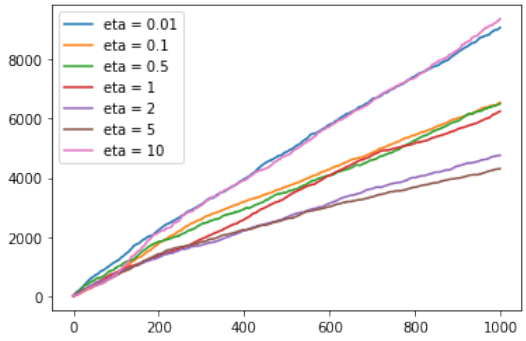

Figure 1. Plot of Regret versus time-step for Bandit OGD with different values of η

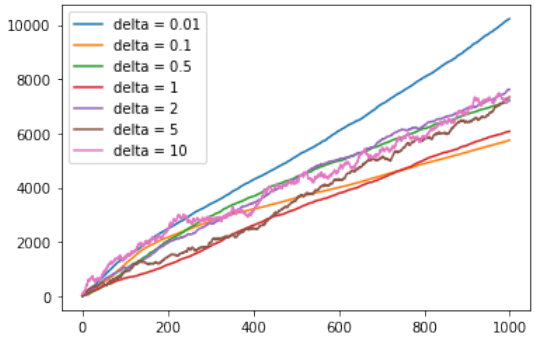

Figure 2. Plot of Regret versus time-step for Bandit OGD with different values of δ

2) Adversary Parameter Sensitivity Analysis:: When we vary the number of cost functions available to the adversary and analyze the change in regret we see that the regret achieved increases as we increase the number of cost functions. This to be expected as the Adversary has more options as we increase the number of cost functions available to it making the regret for the player larger. This is shown in Fig 3 with almost monotonic increase in regret on increasing number of cost function available.

3) Comparison with Theoretical Bounds:: The Theoretical Bound given for this algorithm given in [1] is $O(T^{\frac{3}{4}})$. Running the above algorithm for $\eta = 0.5$, $\delta = 1$ we get the plot given in and comparing with the theoretical Bounds.

B. Bandit Online Gradient Descent without Gradient:

1) Parameter Sensitivity Analysis:

a) Variation with δ : The δ parameter is used to control the update of the vector $\mathbf{w}$ which player chooses. As expected if the value of δ is small the regret tends to get saturated and if large the regret blows up. Therefore it is important to set the optimal value which is found to be between 0.5 and 0.6 which can be observed from Fig 5. Set $\delta \in [0.5, 0.6]$.

b) Variation with α : α is a very sensitive parameter (evident from Fig 6) since regret blows up quickly when it becomes larger than the optimal parameter and wrong bound

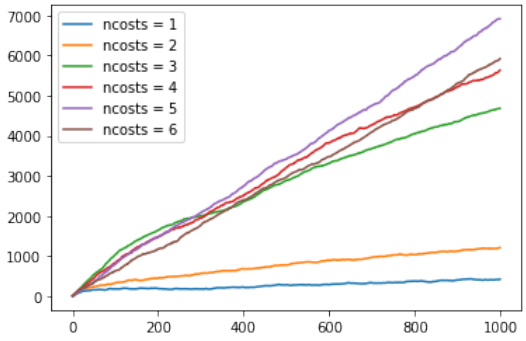

Figure 3. Plot of Regret versus time-step for Bandit OGD with different values of ncosts which is the number of loss functions available to the adversary

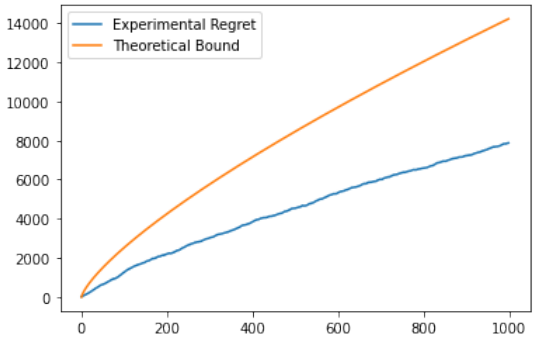

Figure 4. Plot of Regret versus time-step for Bandit OGD with different values of δ

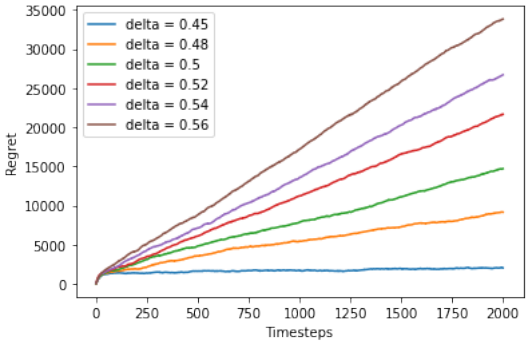

Figure 5. Plot of Regret versus time-step with varying values of δ

if α is less than optimal. This is mainly because α defines the set on which the projection of $\mathbf{w_t} - \nu * c_t * u_t$ is taken.

c) Variation with ν : ν is a insensitive parameter as evident from the Fig 7. As long as the value of δ and α are chosen correctly the value of $\nu \in (0, 1)$ given decent performance.

2) Comparison with Theoretical bounds:: We ran the algorithm for optimal parameters $\alpha = 0.9$, $\delta = 0.5$ and $\nu = 0.2$. The Theoretical bound given in [2] is $O(T^{\frac{5}{6}})$. For finding the observed bound we did linear regression on a log-scale

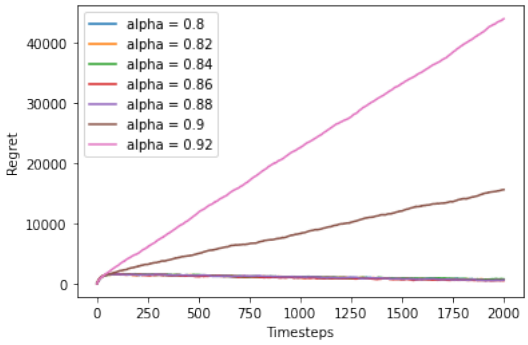

Figure 6. Plot of Regret versus time-step with varying values of α

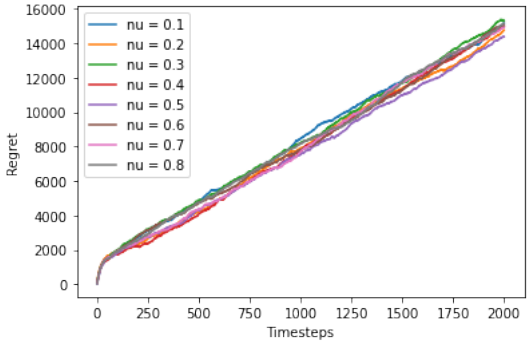

Figure 7. Plot of Regret versus time-step with varying values of ν

and found out the slope to be 0.66 a much better bound than 0.833. This is because we are working with linear functions. Therefore the more tight bound is $O(T^{\frac{3}{4}})$.(Refer Fig 8)

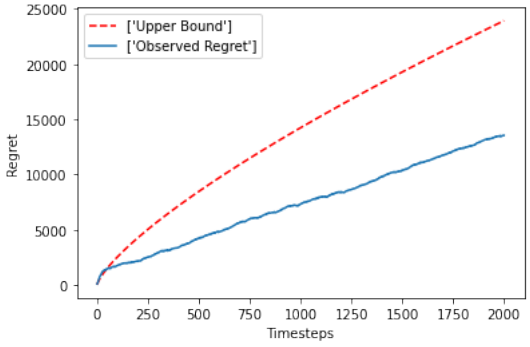

Figure 8. Plot of Regret versus time-step to compare theoretical bound

C. SCRIBLE

1) Parameter Sensitivity Analysis:

a) Variation with η: When looking at senstivity of algorithm with η we see that the algorithm gives the best results for intermediate values of η from 0.5-1.5. For higher or lower values we have relatively poorer performance. This is shown in Fig 9. Although, the variation with η is somewhat noisy this is primarily due to the random nature of the Algorithm.

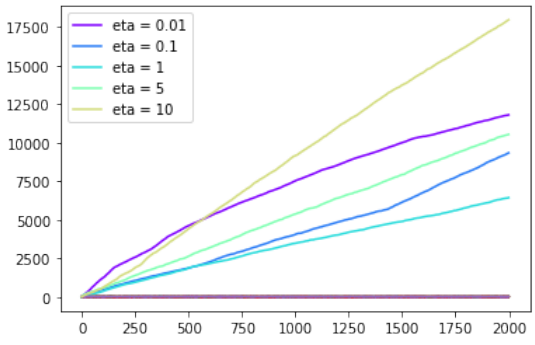

Figure 9. Plot of Regret versus time-step for SCRIBLE Algorithm with different values of η

2) Adversary Parameter Senstivity: When we vary the number of cost functions available to the adversary and analyze the change in regret we see that the regret achieved increases as we increase the number of cost functions. This is the same as other algorithms and is to be expected. This is shown in Fig 10 where we have almost monotonic increase in Regret.

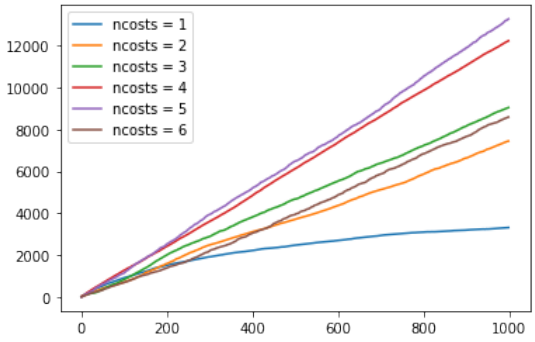

Figure 10. Plot of Regret versus time-step for SCRIBLE Algorithm with different values of ncosts which is the number of loss functions available to the adversary

3) Comparison With Theoretical Bounds: The Theoretical upper bound for this Algorithm is given by $O(\sqrt{T}\log T)$. Running the Algorithm for $\eta = 1$ and comparing with Experimental regret we get the plot in .

D. Bandit Online Linear Optimization

This is an extension of the Algorithm 3, where instead of using the inverse of the Hessian of the self-concordant barrier we use the eigenvectors and eigenvalues of the Hessian.

1) Parameter Sensitivity Analysis:

a) Variation with η: The algorithm is fairly sensitive to the parameter η. As it is increased the regret get more and more bounded. This is due to the reason because η is used to control the update of the vector chosen by the player. Therefore a more optimal vector is chosen because of η.

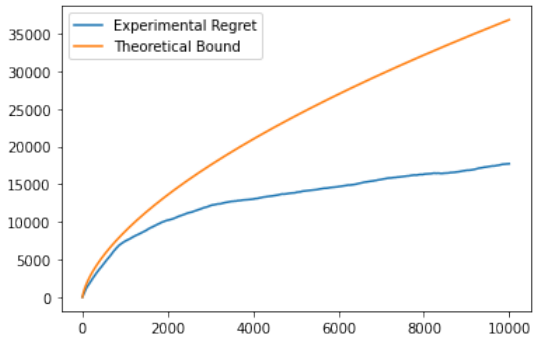

Figure 11. Plot of Regret versus time-step for the SCRIBBLE Algorithm as well as the theoretical bound

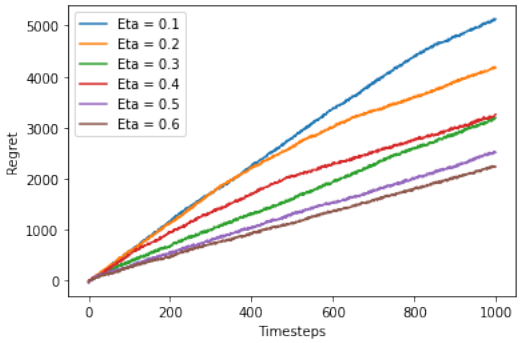

Figure 12. Plot of Regret versus time-step for Algorithm 4 with different values of η

2) Adversary Parameter Sensitivity:: Like all other algorithms as the number of cost functions increases the regret also increases since adversary get more number of options. However for this Algorithm we can see from the figure that the increasing effect gets saturated .

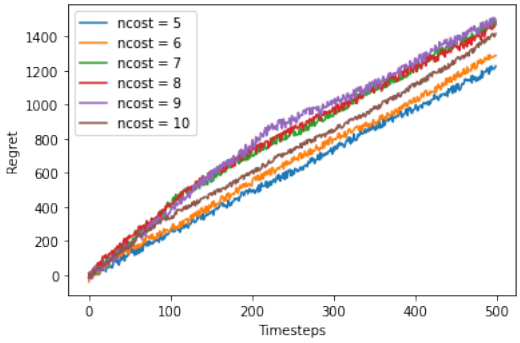

Figure 13. Plot of Regret versus time-step for Algorithm 4 with different values of ncosts

3) Comparison With Theoretical Bounds:: Running the algorithm with $\eta = 0.5$ and time-steps = 1000 the regret is tightly bounded by $O(\sqrt{Tlog(T)})$. (Refer Fig 14) This is in agreement with the bound for regret given in [4].

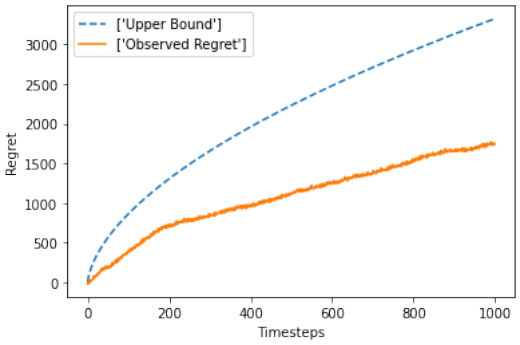

Figure 14. Plot of Regret versus time-step for Algorithm 4 to compare theoretical bound

V. Comparison

A. Comparison

1) Comparison based on Regret Achieved: The regret is best bounded by Algorithm 4 which is given by $O(\sqrt{Tlog(T)}$ because of its use of Interior point methods and eigen decomposition instead of inverse of Hessian. Algorithm 3 comes next with almost same bound of $O(\sqrt{T}log(T))$. Algorithm 2 has a bound of $O(T^{\frac{5}{6}})$ for a general convex function. Because we have limited ourselves to linear functions we get a better bound of $O(T^{\frac{3}{4}})$. Algorithm 1 also has the same bound of $O(T^{\frac{3}{4}})$. Since the only difference between them is that we are using projection on the set for Algorithm 2.

2) Comparison based on Running time: Comparing the running time for each algorithm we find that Algorithm 2 Bandit Online Gradient Descent without A Gradient is the fastest out of the four due to it's relatively simple nature. Followed by Algorithm 2, Algorithm 1 Bandit Online Gradient Descent has a relatively fast running time which is very close to the run time of Algorithm 1. Algorithm 4 follows next which has a significantly higher running time due to the time needed to calculate the Eigen-Vectors and Eigen-Values of the Hessian which are needed for this Algorithm. Algorithm 3 has the highest running time due to two reasons. First the time taken to invert the Hessian which is typically of the order of $O(D^3)$ and the time taken for the minimization step which involves minimizing the objective which contains a logarithmic function.

3) Comparison Based on Number of parameters: Algorithm - 2 has the most parameters δ which controls the update of the vector, η which acts as the learning rate and α which defines the set S on which the vector is projected. Therefore this algorithm gives the most flexibility in terms of choices on how we want the algorithm to run. Algorithm - 1 has two parameters η which is the learning rate and δ which is used to control the update of the vector.
In Algorithm 3 we have to calculate the inverse of the Hessian which is most time taking step which can be overcome in

Algorithm-4 which uses eigenvalues and eigenvectors which can be calculated using power method [7]. We have chosen a $-\sum_i log(x_i)$ self-concordant barrier so that the computation of our Hessian is not time consuming. Further we can implement Newton-Damped method [4] to reduce its complexity.

VI. Recommendations

If the cost functions available to the adversary are non-linear Algorithm 3 and 4 cannot be used and are strictly for Linear Bandit Optimization. Hence, Algorithm 1 or Algorithm 2 should be used instead based on other parameters. If low cost or low regret is the deciding parameter Algorithm 3 or 4 should be used since they achieve the lowest regrets and also have the lowest regret bounds. Algorithm 4 performs better and hence, should be the ideal candidate. If running time is the deciding parameter Algorithm 1 or 2 should be used as they have the lowest running times out of the lot. Algorithm 2 having the lowest time should be preferred. If the deciding parameter is model complexity and flexibility Algorithm 1 or 2 should be used based on the application as they can be used and can be without having to choose a regularization function and can be used generally for more cost functions.

VII. Conclusion

In this report we have done rigorous analysis on the effect the parameters have on their algorithm. We proved the theoretical bounds of regret experimentally by simulating the algorithms for the optimal parameters. We have also shown for which parameters the algorithm may fail. The use of self-concordant barriers indeed improve the regret bound but are time-consuming due computation of Hessian of the self-concordant function. While Algorithm 1 and 2 can be used for any set of convex functions Algorithm 3 and 4 are limited to Linear Optimization.

References

[1] Shai Shalev-Shwartz, "Online Learning and Online Convex Optimization"

[2] Abraham D. Flaxman, Adam Tauman Kalai, H. Brendan McMahan, "Online convex optimization in the bandit setting:gradient descent without a gradient"

[3] Elad Hazan, "Introduction to Online Convex Optimization.,"

[4] Jacob Abernethy,Elad Hazan,Alexander Rakhlin, Competing in the Dark: An Efficient Algorithm for Bandit Linear Optimization

[5] V. MULLACHERY AND S. TIWARI ONLINE BANDIT LINEAR OPTIMIZATION: A STUDY

[6] Pavel Dvurechensky, Petr Ostroukhov, Kamil Safin, Shimrit Shtern, Mathias Staudigl Proceedings of the 37th International Conference on Machine Learning. Self-Concordant Analysis of Frank-Wolfe Algorithms

[7] MAYSUM PANJU, .ITERATIVE METHODS FOR COMPUTING EIGENVALUES AND EIGENVECTORS

VIII. Appendix

A. Link to code

All codes are written by us. Standard procedures for plotting and optimization using cvxpy are used. Link to the Google Colab Notebooks used for each algorithm is given below

Algorithm 1: Bandit Online Gradient Descent
Algorithm 2: Bandit Online Gradient Descent without gradient
Algorithm 3: SCRIBLE - Self Concordant Regularization in Bandit Learning
Algorithm 4: Bandit Online Linear Optimization