# An Empirical Analysis of Bandit Convex Optimization Algorithms

**Naren Manoj** MANOJ.NARENS@GMAIL.COM
**Prabhat Nagarajan** PRABHAT.NAGARAJAN@GMAIL.COM

## Abstract

We perform an empirical analysis of bandit convex optimization (BCO) algorithms. We motivate and introduce multi-armed bandits, and explore the scenario where the player faces an adversary that assigns different losses. In particular, we describe adversaries that assign linear losses as well as general convex losses. We then implement various BCO algorithms in the unconstrained setting and numerically confirm their regret bounds. We find that in the BCO model, despite the lack of a gradient oracle, we can still approximate the gradient reasonably well and thereby attain a minimizer for the losses after a sufficient number of iterations.

## 1. Introduction

Multi-armed bandit (MAB) problems are sequential decision-making problems where actions must be taken at each time step, where an action corresponds to drawing a reward from a distribution. The player's goal is to maximize the total reward over some sequence of steps. This problem formulation has a wide variety of practical applications (Bubeck et al., 2012). For example, ad placement: the problem of selecting an advertisement for the next user of a website can be modeled as a MAB problem. We can view the ad placement as the action and the reward being generated by a user clicking on an ad. Furthermore, source routing in a network can be modeled as a MAB, where the reward is a function of the time taken to deliver the network packets. Lastly, in the game of Go, MABs have been used to improve the efficiency of the tree search of game continuations.

Bandit Convex Optimization (BCO) is a special case Online Convex Optimization (Agarwal et al., 2010). In BCO, a player seeks to minimize a sequence of loss functions generated by an adversary while only observing the value of each function. As the player performs multiple actions, he has incomplete information. That is, he only has the information obtained from observing the value generated as a consequence of his action. As such, the player experiences a tradeoff between *exploiting* the information he has gathered while *exploring* other actions to gain information that can later be exploited. Naturally, in this scenario, we would like to consider the player's objective. Formally, the player's goal is to minimize his *regret* (Bubeck et al., 2012). We can find the regret by comparing the player's performance to that of an optimal player that performs the correct action at each step.

Our motivation is to run an empirical study on certain BCO algorithms, and observe several performance parameters. Understanding how these algorithms perform under different types of loss functions can provide insights into the effectiveness of these algorithms in practice. Furthermore, to our knowledge, there has not been extensive experimentation of these algorithms in the BCO literature.

The outline of this report is as follows. In the following section, we provide a formal background on multi-armed bandits and BCO. We then perform experiments on online gradient descent and present our findings.

## 2. Background

### 2.1. Multi-armed Bandits

**Definition 1.** *The $K$-**armed bandit** problem (Auer et al., 2002) consists of random variables $X_{i,n}$ for $i \in [1, ..., K]$, $K \geq 2$, $n \geq 1$, where $n$ is the number of time steps. The index $i$ refers to the index of a "gambling machine" (i.e. an arm). The sequence $X_{i,1}, X_{i,2}, ..., X_{i,n}$ are the rewards if the machine $i$ is played $n$ times, where $X_{i,k}$, $k \in [1, ..., n]$ are independently and identically distributed.*

The standard analogy of the $K$-armed bandit problem is to $K$ slot machines at a casino. Each slot machine yields a reward drawn from a distribution corresponding to that machine. For $n$ time steps, the player is allowed to play one slot machine at each time step, with the goal of maximizing his total reward. The player does not observe what would have occurred had he selected a different machine at that time step.

We present the multi-armed bandit problem as a precursor. The definition presented here is in its simplest form.

**Definition 2.** *The **regret** after $n$ plays $I_1, ..., I_n \in \{1, ..., K\}$ is defined as:*

$$R_n = \max_{i=1,\dots,K} \sum_{t=1}^{n} X_{i,t} - \sum_{t=1}^{n} X_{I_t,t} \tag{1}$$

In other words, the regret over $n$ time steps is the sum of the differences between the reward received when playing optimally and what the player receives at each time $t$.

**Definition 3.** *The **expected regret** after $n$ plays $I_1, ..., I_n \in \{1, ..., K\}$ is defined as:*

$$\mathbb{E}\left[R_n\right] = \mathbb{E}\left[\max_{i=1,\dots,K} \sum_{t=1}^{n} X_{i,t} - \sum_{t=1}^{n} X_{I_t,t}\right] \tag{2}$$

## 2.2. Bandit Convex Optimization

We now define the **Adversarial Bandit Problem**, following the notation from (Bubeck et al., 2012).

**The Adversarial Bandit Problem**
*Givens:* $K \geq 2$. Number of rounds $n \geq K$
For $t = 1, ..., n$

1. The player selects an arm $I_t \in \{1, ..., K\}$
2. Simultaneously, the adversary chooses a gain vector $g_t = (g_{1,t}, ..., g_{k,t}) \in [0, 1]^K$
3. The player receives reward $g_{I_t,t}$

## 2.3. Linear Optimization

We can generalize the setting of the Adversarial Bandit Problem. We replace the set of arms $\{1, .., K\}$ by $\mathcal{K} \subset \mathbb{R}^d$ where $\mathcal{K}$ is compact. There is a loss function at each time step that is defined on $\mathcal{K}$. The player must choose an arm (or a point in $\mathcal{K}$) that minimizes the loss (Bubeck et al., 2012). In this scenario, we assume that the loss function at each time step is linear over $\mathcal{K}$. This assumption is reasonable since many practical problems can modeled with a linear loss, such as the source routing problem from the introduction. The player selects an "arm" $x_t \in \mathcal{K}$. The adversary selects $l_t \in \mathcal{L} \subset \mathbb{R}^d$. The resultant loss is $\langle x_t, l_t \rangle$, a linear function of $x_t$.

The linear optimization algorithm we use is outlined in algorithm 1, and comes from (Abernethy et al., 2008). We repeat their algorithm (using the same notation).

**Algorithm 1** Bandit Online Linear Optimization

```
1: Input: $\eta > 0$, $\nu$-self concordant $\mathcal{R}$.
2: $x_1 = \arg\min_{x \in \mathcal{K}} [\mathcal{R}(\mathbf{x})]$.
3: for $t = 1$ to $T$ do
4:     Let $\{\mathbf{e}_1, ..., \mathbf{e}_n\}$ and $\{\lambda_1, ..., \lambda_k\}$ be the eigenvectors and eigenvalues of $\nabla^2 \mathcal{R}(x_t)$.
5:     Choose $i_t$ normally uniformly at random from $\{1, ..., n\}$ and $\epsilon_t = \pm 1$ with probability $\frac{1}{2}$
6:     Predict $\mathbf{y}_t = \mathbf{x}_t + \epsilon_t \lambda_{i_t}^{-1/2} \mathbf{e}_{i_t}$
7:     Observe the gain $\mathbf{f}_t^T \mathbf{y}_t \in \mathbb{R}$.
8:     Define $\tilde{\mathbf{f}}_t := n(\mathbf{f}_t^T \mathbf{y}_t) \epsilon_t \lambda_{i_t}^{1/2} \cdot \mathbf{e}_{i_t}$
9:     Update
           $\mathbf{x}_{t+1} = \arg\min_{x \in \mathcal{K}} \left[ \eta \sum_{s=1}^{t} \tilde{\mathbf{f}}_s^t \mathbf{x} + \mathcal{R}(\mathbf{x}) \right]$
10: end for
```

## 2.4. Gradient Estimation

We now generalize our game to allow for nonlinear but convex losses. At each step, we receive feedback on $k$ points and receive a penalty on one of these points at random. Recall that our goal is to minimize our regret over iterations; that is, at each iteration, we want to reach an optimal constant point over all of the losses we have seen previously.

Since we do not receive gradient information at each step in our game, it is necessary to devise some sort of gradient estimation. Here, we summarize known techniques for estimating gradients. Each of these estimators develops an unbiased estimate of the gradient of $\hat{f}$, a smoothed modification of the original function $f$. Before we begin, we state the below notation (Bubeck et al., 2012; Agarwal et al., 2010):

- Let $\mathbb{B}^d$ denote the unit Euclidean ball in $\mathbb{R}^d$. Mathematically, we can state:

$$\mathbb{B}^d = \{v \in \mathbb{R}^d \ : \ \|v\|_2 \leq 1\}$$

- Similarly, let $\mathbb{S}^d$ denote the unit Euclidean sphere in $\mathbb{R}^d$. This is equivalent to writing:

$$\mathbb{S}^d = \{v \in \mathbb{R}^d \ : \ \|v\|_2 = 1\}$$

### 2.4.1. Spherical Estimator

As its name suggests, the spherical estimator samples points from a sphere to obtain an unbiased estimate of the gradient of a smoothed version of $f$. Specifically, for some parameter $\delta > 0$, we define $\hat{f}$ as follows:

$$\hat{f}(x) := \mathop{\mathbb{E}}_{v \sim \mathbb{B}^n} \left[ f(x + v\delta) \right]$$

We have the following theorem (Agarwal et al., 2010):

**Theorem 4.** *Suppose $u \sim \mathbb{S}^n$. Then,*

$$\nabla \hat{f}(x) = \mathop{\mathbb{E}}_{u} \left[ \frac{n}{\delta} f(x + u\delta) u \right]$$

To generate random samples from $\mathbb{S}^d$, observe that every point on $\mathbb{S}^d$ can be uniquely expressed in spherical coordinates. Furthermore, note that spherical coordinates uniquely maps to points on $\mathbb{S}^d$. As a result, it suffices to select $d-1$ angles uniformly at random and convert this set of angles from spherical coordinates to a point in rectangular coordinates. We introduce an algorithm that generates these samples, Algorithm 2.

**Algorithm 2** Generate samples from $\mathbb{S}^d$

```
1: Input: {}
2: $p \leftarrow \{0\}^d$
3: $\theta \leftarrow \{\theta_1, \dots, \theta_{d-1}\}$ where $\theta_i \sim [0, 2\pi]$ uniformly
4: for $1 \le i \le d-1$ do
5:     $p_i = \cos(\theta_i) \prod_{j=1}^{i-1} \sin(\theta_j)$
6: end for
7: $p_d = \prod_{j=1}^{d-1} \sin(\theta_j)$
8: Output: $p$
```

We can use this algorithm to generate samples from $\mathbb{B}^d$ as well. All that remains is to select a magnitude for our point uniformly at random from $[0, 1]$. A pseudocode implementation of this is detailed in Algorithm 3.

**Algorithm 3** Generate samples from $\mathbb{B}^d$

```
1: Input: {}
2: $p \leftarrow$ output from Algorithm 2
3: $r \leftarrow v$ where $v \sim [0, 1]$ uniformly
4: $p \leftarrow pv$
5: Output: $p$
```

Using these subroutines, we can implement our gradient estimation.

### 2.4.2. Ellipsoidal Estimator

We can generalize the previous estimator to incorporate samples from an ellipsoid instead of a sphere. For some matrix $A \in \mathbb{R}^{n \times n}$ and $v \sim \mathbb{B}^n$, we redefine $\hat{f}(x)$:

$$\hat{f}(x) := \mathbb{E}\left[ f(x + Av) \right]$$

Now, we provide the below theorem (Agarwal et al., 2010):

**Theorem 5.** *If $u \sim \mathbb{S}^n$, then we have:*

$$\nabla \hat{f}(x) = \mathbb{E}\left[ n f(x + Au) A^{-1} u \right]$$

Observe that an implementation of this estimator is as straightforward as an implementation of the previous estimator.

### 2.4.3. One and Two Point Estimates

Now, given this information and one or two points of feedback, we can prove bounds on the quality of the gradient estimates obtained using the aforementioned theorems. First, we state bounds on the $l^2$-norm of the gradient estimates.

**Theorem 6.** *(Bubeck et al., 2012) Suppose that at each round in our optimization process, we receive one point of feedback. Then, for $u \sim \mathbb{S}^d$ uniformly and for some exploration parameter $\delta$, if we set:*

$$g_t := \frac{d}{\delta} f_t(x + u\delta) u$$

*we have:*

$$\|g_t\|_2 \le \frac{dL}{\delta}$$

*if $f_t(x) \le L$.*

**Theorem 7.** *(Bubeck et al., 2012) Suppose that at each round in our optimization process, we receive two points of feedback. Then, for $u \sim \mathbb{S}^d$ uniformly and for some exploration parameter $\delta$, if we set:*

$$g_t = \frac{d}{2\delta} \left( f_t(x + u\delta) - f_t(x - u\delta) \right) u$$

*we have:*

$$\|g_t\|_2 \le Gd$$

*if $f_t$ is $G$-Lipschitz.*

Notice that there is no dependence on $\delta$ when we use a two-point feedback setting as opposed to a one-point setting. This key difference allows us to obtain significantly better regret bounds in the two-point setting than in the one-point setting. We analyze these further in the following section.

## 2.5. Online Gradient Descent with Multi-Point Feedback

Consider Algorithm 4, adapted from (Agarwal et al., 2010). Observe that this algorithm follows the same general structure as projected gradient descent. However, notice that this algorithm has one key difference from projected gradient descent —namely, instead of receiving gradient feedback, we receive feedback from some loss function.

**Algorithm 4** General Online Bandit Projected Gradient Descent

```
1: Input: Closed and convex set $\mathcal{X} \subseteq \mathbb{R}^d$, step size $\eta$, number of iterations $T$, amount of feedback per round $k$
2: $x_0 \leftarrow \{0\}^d$
3: $r \leftarrow 0$
4: for $1 \leq t \leq T$ do
5:     Adversary chooses convex loss $f_t$
6:     $P \leftarrow \{p_1, \dots, p_k\}$ based on $x_{t-1}$ somehow
7:     $z \leftarrow y$ where $y \sim P$ uniformly
8:     $F_t \leftarrow \{f_t(p_1), \dots, f_t(p_k)\}$
9:     $g_t \leftarrow$ gradient estimate using $F_t$
10:    $x_t \leftarrow \Pi_{\mathcal{X}}(z - \eta g_t)$
11:    $r \leftarrow r + f_t(z)$
12: end for
13: $b \leftarrow \min_{x' \in \mathcal{X}} \sum_{t=1}^{T} f_t(x')$
14: $r \leftarrow r - b$
15: Output: $x_T$, $r$ (computed optimum point and total regret, respectively)
```

We will now briefly discuss some theoretical properties of this algorithm. In particular, we will consider this algorithm when we receive one and two points of feedback in the unconstrained setting. To start, consider the modification of the generalized algorithm (Algorithm 5 (Bubeck et al., 2012)), which makes use of one point of feedback.

**Algorithm 5** One Point Feedback OSGD

```
1: Input: Step size $\eta$, number of iterations $T$, exploration parameter $\delta$
2: $x_0 \leftarrow \{0\}^d$
3: $r \leftarrow 0$
4: for $1 \leq t \leq T$ do
5:     Adversary chooses convex loss $f_t$
6:     $u \leftarrow v$ where $v \sim \mathbb{S}^d$
7:     $y_t \leftarrow x_{t-1} + u\delta$
8:     Player plays $y_t$ and observes loss $f_t(y_t)$
9:     $g_t \leftarrow \frac{d}{\delta} f_t(y_t) u$
10:    $x_t \leftarrow x_{t-1} - \eta g_t$
11:    $r \leftarrow r + f_t(y_t)$
12: end for
13: $b \leftarrow \min_{x' \in \mathcal{X}} \sum_{t=1}^{T} f_t(x')$
14: $r \leftarrow r - b$
15: Output: $x_T$, $r$ (computed optimum point and total regret, respectively)
```

Similarly, consider Algorithm 6 (Bubeck et al., 2012), which makes use of two points of feedback.

**Algorithm 6** Two Point Feedback OSGD

```
1: Input: Step size $\eta$, number of iterations $T$, exploration parameter $\delta$
2: $x_0 \leftarrow \{0\}^d$
3: $r \leftarrow 0$
4: for $1 \leq t \leq T$ do
5:     Adversary chooses convex loss $f_t$
6:     $u \leftarrow v$ where $v \sim \mathbb{S}^d$ uniformly
7:     $y_t^+ \leftarrow x_{t-1} + u\delta$
8:     $y_t^- \leftarrow x_{t-1} - u\delta$
9:     $y_t \leftarrow y_t^+$ with probability $\frac{1}{2}$ and $y_t^-$ otherwise
10:    Player plays $y_t$ and observes loss $f_t(y_t)$
11:    Player observes $f_t(y_t^+)$ and $f_t(y_t^-)$
12:    $g_t \leftarrow \frac{d}{2\delta}\left(f_t(y_t^+) - f_t(y_t^-)\right) u$
13:    $x_t \leftarrow x_{t-1} - \eta g_t$
14:    $r \leftarrow r + f_t(y_t)$
15: end for
16: $b \leftarrow \min_{x' \in \mathcal{X}} \sum_{t=1}^{T} f_t(x')$
17: $r \leftarrow r - b$
18: Output: $x_T$, $r$ (computed optimum point and total regret, respectively)
```

We now state the following theorems regarding the regret for Algorithm 5 and Algorithm 6, taken from (Bubeck et al., 2012).

**Theorem 8.** *If we take $n$ steps of Algorithm 6, then our regret is $O(\sqrt{n})$.*

**Theorem 9.** *If we take $n$ steps of Algorithm 5, then our regret is $O\left(n^{3/4}\right)$.*

In our experiments, we will observe these regret bounds.

## 3. Experiments

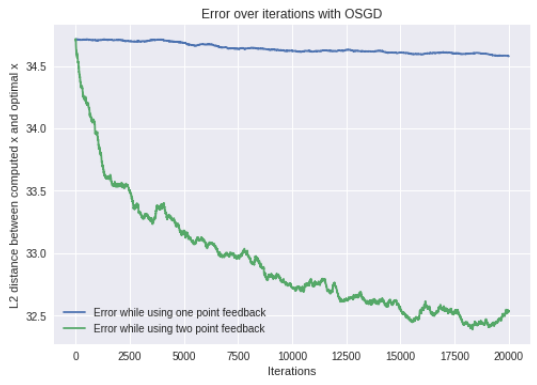
We explored the numerical performance of OSGD algorithms.

Our experiments were based on the canonical $l^1$-regularized least squares problem. Specifically, suppose we have a vector $x \in \mathbb{R}^d$ that we would like to estimate. We have $m$ measurements of $y = A_i^T x + b$, where $b$ is small random noise. In our experiments we consider the cases where $m < d$ (the under-determined, non-strongly convex case) as well as the case where $m \geq d$ (over-determined).

To motivate using BCO for this problem, suppose that we are in a setting where we have far too many measurements to realistically load into memory at once. As a result, it is impractical to compute full gradient updates or full function evaluations. Furthermore, suppose we do not have access to a gradient oracle; instead, we have access to a function evaluation over some subset of the observations. Such a model is conceivable since, intuitively, it is far more likely for one to have access to a function oracle rather than a gradient oracle.

It therefore follows that we can use BCO under such a problem setting. The adversary can select any of $\binom{m}{b}$ loss functions where $b$ is the maximum amount of observations that we would like to load into memory at once. Since we do not have a gradient oracle, we must use one of the gradient estimators we discussed earlier in order to determine the direction our solution path takes. A full implementation of this can be found in the code [1] [2].

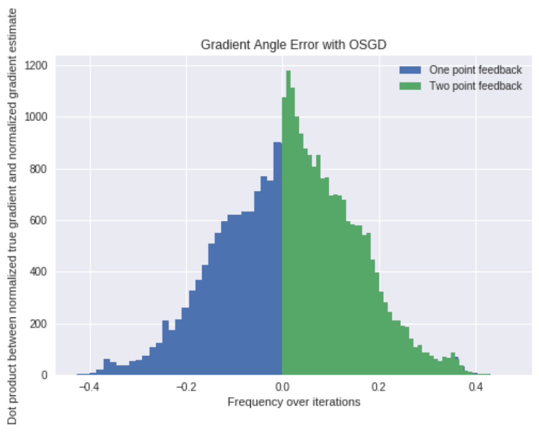
This IPython notebook contains a guide to modify the hyperparameters to obtain more results. We attach a subset of our experimental results here along with a caption describing each experiment. In the following graphs, we used $m = 20$ (undetermined). Each entry in $A$ was i.i.d. according to a standard normal distribution. Then, each observation was perturbed with small Gaussian noise.



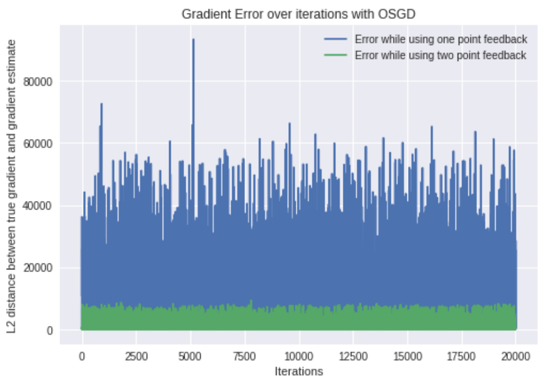
Measures $\|x - x_{\text{true}}\|_2$ over our iterations.

[1] narenmanoj.github.io/ee381k_final_project_code.ipynb

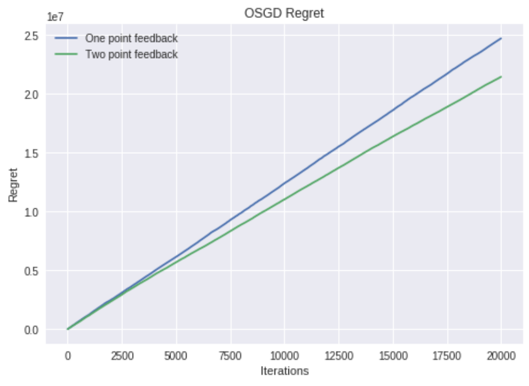
[2] prabhatnagarajan.github.io/docs/final_project.ipynb



Measures the distribution of dot products between the normalized gradient and the normalized gradient estimate. We collected this data in order to see how often the gradient for each method misleads the player.



Measures $\|g_t - \tilde{g}_t\|_2$ over our iterations.



Measures the regret over our iterations.

Observe that each of these graphs suggests that the two point model is far stronger than the one point model. Specifically, note that the gradient estimate always has a component parallel to the true gradient at that point. This implies that we might never move *away* from the optimal solution (even though we might tread a very elongated solution path). We can formally state as a conjecture.

**Conjecture 10.** *In Algorithm 6, if $\tilde{g}_t$ is the gradient estimate at time $t$ and $g_t$ is the true gradient, then:*

$$\tilde{g}_t^T g_t \geq 0$$

Additionally, our results seem to suggest that the following is true.

**Conjecture 11.** *In Algorithm 5, if $\tilde{g}_t$ is the gradient estimate at time $t$ and $g_t$ is the true gradient, then:*

$$\mathbb{E}\left[\tilde{g}_t^T g_t\right] = 0$$

Another interesting direction involves obtaining concentration results on the gradient estimation. In particular, if we can show that the normalized gradient estimate does not deviate too far from the normalized true gradient at each step with high probability, then such a proof may lend insight into how these gradient estimators work or how to devise potentially better gradient estimators.

We noticed that in the one-point model, we obtained better convergence results while using a higher value of $\delta$. This is intuitively true as a result of Theorem 6. In particular, it becomes more difficult to bound the norm of the gradient estimate as $\delta$ increases. However, while inceasing $\delta$ allows us to obtain better gradient estimates, we also have smaller movements along the solution path at each timestep. As a result, convergence could be quite slow.

This tradeoff behaves differently in the two-point model. Since Theorem 7 does not have any dependence on $\delta$, we can make $\delta$ very small. However, as we increased $\delta$, we found that we could attain larger movements at each timestep while the distance between the current solution and the optimum behaved with seemingly no pattern.

## 4. Conclusion

In this project, we described and explored BCO algorithms. We restated central theoretical results from the literature and implemented various BCO algorithms. We then analyzed the results from our experiments. In the future, we would like to try proving the two conjectures introduced towards the end of our analysis. We would also like to conduct more experiments using different loss functions to see whether we can further observe interesting behavior in these algorithms. The ideal scenario involves making theoretical conjectures inspired by experimental results and proving them.

## References

Abernethy, Jacob D, Hazan, Elad, and Rakhlin, Alexander. Competing in the dark: An efficient algorithm for bandit linear optimization. In *COLT*, pp. 263–274, 2008.

Agarwal, Alekh, Dekel, Ofer, and Xiao, Lin. Optimal algorithms for online convex optimization with multi-point bandit feedback. In *COLT*, pp. 28–40, 2010.

Auer, Peter, Cesa-Bianchi, Nicolo, and Fischer, Paul. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

Bubeck, Sébastien, Cesa-Bianchi, Nicolo, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.