1 Bandit Convex Optimization

2 Multi-Armed Bandit Optimization

3 Stochastic Multi-Armed Bandit Optimization
At each iteration $t$, the player chooses $x_t$ in \textit{convex} set $\mathcal{K}$.

A \textit{convex} loss function $f_t \in \mathcal{F} : \mathcal{K} \to \mathbb{R}$ is revealed.

A cost $f_t(x_t)$ is incurred.

\begin{itemize}
  \item $\mathcal{F}$ is a set of bounded functions.
  \item $f_t$ is revealed after choosing $x_t$.
  \item $f_t$ can be adversarially chosen.
\end{itemize}
Recap: Online Convex Optimization

Goal: minimize the regret bound

$$\text{regret}_T = \sum_{t=1}^{T} f_t(x_t) - \min_{x \in \mathcal{K}} \sum_{t=1}^{T} f_t(x)$$

Online Gradient Descent (OGD) (Zinkevich 2003):

$$x_{k+1} = \Pi_{\mathcal{K}}(x_k - \eta_t \nabla f_t(x_t))$$

Regret bound

- if $f_t$ is convex: $O(GD \sqrt{T})$
- if $f_t$ is $\alpha$-strongly convex: $O\left(\frac{G^2}{2\alpha} (1 + \log(T))\right)$
Motivation

- In Ad-placement, the search engine can inspect which ads were clicked through, but cannot know whether different ads would have been click through or not.
- Given a fixed budget, how to allocate resources among the research projects whose outcome is only partially known at the time of allocation and may change through time.
Bandit Convex Optimization

Motivation

- In Ad-placement, the search engine can inspect which ads were clicked through, but cannot know whether different ads would have been click through or not.
- Given a fixed budget, how to allocate resources among the research projects whose outcome is only partially known at the time of allocation and may change through time.

Bandit Setting

- In OCO, player has access to $\nabla f_t(x_t)$
- In BCO, player only has black-box access to the function value $f_t(x_t)$. We only can evaluate each function once.
**Exploration vs Exploitation**

*Balance* between exploiting the gathered information and exploring the new data.

*Figure*: Where to eat? (Image source: UC Berkeley AI course slide, lecture 11.)
OGD without a gradient

**Question:** Can we perform OGD without gradients?

- One dim

\[ \tilde{\nabla} f(x) = \frac{(f(x + \delta) - f(x - \delta))}{2\delta} \]
**OGD without a gradient**

**Question:** Can we perform OGD without gradients?

- **One dim**
  \[ \tilde{\nabla} f(x) = \frac{(f(x + \delta) - f(x - \delta))}{2\delta} \]

- **d dim**
  \[ \tilde{\nabla} f(x) \approx \mathbb{E}_{u \in \partial B}[(f(x + \delta u) - f(x))u]d/\delta \]
  \[ = \mathbb{E}_{u \in \partial B}[f(x + \delta u)u]d/\delta \]

**Note:** \( \tilde{g}(x, u) = f(x + \delta u)ud/\delta \)

\[ \mathbb{E}_{u \in \partial B}[\tilde{g}(x, u)] = \nabla \hat{f}(x), \text{ with } \hat{f}(x) = \mathbb{E}_{v \in B}[f(x + \delta v)] \]
Bandit gradient descent algorithm

**Assumption:**

- only access to $f_t$ at one single point $x_t$.
- function value is bounded, $\{f_t\} : \mathcal{K} \to [-C, C]$.
- $f_t$ can be non-smooth, no bounded gradient assumption.
- $\exists r, R > 0, r\mathbb{B} \subset \mathcal{K} \subset R\mathbb{B}$. 
Bandit gradient descent algorithm

Assumption:

- only access to $f_t$ at one single point $x_t$.
- function value is bounded, $\{f_t\} : \mathcal{K} \rightarrow [-C, C]$.
- $f_t$ can be non-smooth, no bounded gradient assumption.
- $\exists r, R > 0, r\mathbb{B} \subset \mathcal{K} \subset R\mathbb{B}$.

Algorithm (Flaxman et al. 2005)

- Let $y_1 = 0$, learning rate $\eta, \xi \in (0, 1), \delta > 0$
- for $t = 1, \ldots, T$:
  - select $u_t \in \partial\mathbb{B}$ uniformly at random
  - $x_t = y_t + \delta u_t$ and receive $f_t(x_t)$
  - $y_{t+1} = \Pi_{(1-\xi)\mathcal{K}}(y_t - \eta f_t(x_t)u_t d/\delta)$
    $(y_{t+1} \in (1-\xi)\mathcal{K}$ ensures $x_t \in \mathcal{K}$ for any $\delta \in [0, \xi r])$
For sufficient large $T$ with $\eta = \frac{R}{C\sqrt{T}}$, the expected regret bound is

$$\mathbb{E}\left[\sum_{t=1}^{T} f_t(x_t)\right] - \min_{x \in \mathcal{K}} \sum_{t=1}^{T} f_t(x) \leq 6T^{5/6}dC$$

With additional assumption $L$-Lipschitz function

$$\mathbb{E}\left[\sum_{t=1}^{T} f_t(x_t)\right] - \min_{x \in \mathcal{K}} \sum_{t=1}^{T} f_t(x) \leq 6T^{3/4}d(\sqrt{CLR} + C)$$

Parameters: $T > \left(\frac{3Rd}{2r}\right)^2$, $\delta = \left(\frac{rR^2d^2}{12T}\right)^{1/3} \leq \xi r$, and $\xi = \left(\frac{3Rd}{2r\sqrt{T}}\right)^{1/3}$
Multi-Point Bandit Feedback

Recall

\[ \tilde{g}_t = \frac{d}{\delta} f_t(u_t) u_t \] with \[ \| \tilde{g}_t \| \leq \frac{dC}{\delta} \]
Recall

\[ \tilde{g}_t = \frac{d}{\delta} f_t(u_t) u_t \quad \text{with} \quad \|\tilde{g}_t\| \leq \frac{dC}{\delta} \]

Multi-point scheme (Agarwal et al. 2010): use two function values to construct bounded norm gradient estimators for \textit{L}-Lipschitz continuous functions.

\[ \tilde{g}_t = \frac{d}{2\delta} (f_t(x_t + \delta u_t) - f_t(x_t - \delta u_t)) u_t \quad \text{with} \quad \|\tilde{g}_t\| \leq Ld \]
Recall

\[ \tilde{g}_t = \frac{d}{\delta} f_t(u_t) u_t \text{ with } \|\tilde{g}_t\| \leq \frac{dC}{\delta} \]

Multi-point scheme (Agarwal et al. 2010): use two function values to construct bounded norm gradient estimators for \(L\)-Lipschitz continuous functions.

\[ \tilde{g}_t = \frac{d}{2\delta} (f_t(x_t + \delta u_t) - f_t(x_t - \delta u_t)) u_t \text{ with } \|\tilde{g}_t\| \leq Ld \]

Expected regret bound:

- \( \eta = \frac{1}{\sqrt{T}}, \delta = \frac{\log(T)}{T} \) and \( \xi = \frac{\delta}{r} : (d^2L^2 + R^2)\sqrt{T} + L\log(T)(3 + \frac{R}{r}) \)

- \( \alpha \)-strong convex, \( \eta_t = \frac{1}{\alpha t}, \delta = \frac{\log(T)}{T} \) and \( \xi = \frac{\delta}{r} : L\log(T)\left(\frac{d^2L}{\alpha} + \frac{R}{r} + 3\right) \).
### Summary on regret bounds

#### Figure: Known regret bounds in the Full-Info./BCO setting (Hazan and Levy 2014)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Convex</th>
<th>Linear $\Theta(\sqrt{T})$</th>
<th>Smooth $\tilde{O}(T^{3/4})$</th>
<th>Str.-Convex $\tilde{O}(\sqrt{T})$</th>
<th>Str.-Convex &amp; Smooth $\Theta(\log T)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full-Info.</td>
<td>$\Omega(\sqrt{T})$</td>
<td>$\tilde{O}(\sqrt{T})$</td>
<td>$\tilde{O}(T^{2/3})$</td>
<td>$\Omega(\sqrt{T})$</td>
<td></td>
</tr>
<tr>
<td>BCO</td>
<td>$\tilde{O}(T^{3/4})$</td>
<td>$\tilde{O}(\sqrt{T})$</td>
<td>$\tilde{O}(T^{2/3})$</td>
<td>$\tilde{O}(\sqrt{T})$ [Thm. 10]</td>
<td></td>
</tr>
</tbody>
</table>
Multi-Armed Bandit

Setting

- At iteration $t$, player chooses action $i_t$ from a set of discrete actions $\{1, \ldots, n\}$.
- A loss in $[0, 1]$ is independently chosen for each action.
- The loss associated with $i_t$ is revealed.
- Various assumptions and constraints.

Example: A gambler pulls one of $n$ slot machines to receive a reward or payoff. Each arm is configured with fixed unknown reward/payoff probability. What is the best strategy to achieve highest long-term rewards/lowest cumulative loss?
Multi-Armed Bandit

Setting

- At iteration $t$, player chooses action $i_t$ from a set of discrete actions $\{1, \ldots, n\}$.
- A loss in $[0, 1]$ is independently chosen for each action.
- The loss associated with $i_t$ is revealed.
- Various assumptions and constraints.

Example

A gambler pulls one of $n$ slot machines to receive a reward or payoff. Each arm is configured with fixed unknown reward/payoff probability.

What is the best strategy to achieve highest long-term rewards/lowest cumulative loss?
**Exploration vs Exploitation**: explore more actions or make the best decision using the current estimates of the loss distribution.

**Algorithms**
- Simple MAB algorithm
- EXP3
Multi-Armed Bandit

**Exploration vs Exploitation:** explore more actions or make the best decision using the current estimates of the loss distribution.

**Algorithms**
- Simple MAB algorithm
- EXP3

Let $\mathcal{K} = \Delta_n$ be an $n$-dimensional simplex. The linear loss function

$$f_t(x_t) = \ell_t^\top x_t = \sum_{i=1}^{n} \ell_t(i)x_t(i) \quad \forall x_t \in \mathcal{K}$$

Key: to estimate gradient $\ell_t$. 

Separating exploration and exploitation steps (Hazan 2016)

Algorithm 1 Simple MAB algorithm

1: $\epsilon \in [0, 1]$, learning rate $\eta > 0$.
2: for $t = 1, \ldots, T$ do
3: \hspace{1em} $b_t \sim \text{Bernoulli}(\epsilon)$.
4: \hspace{1em} if $b_t = 1$ then
5: \hspace{2em} Choose $i_t$ uniformly at random and receive $\ell_t(i_t)$
6: \hspace{2em} Let
7: \hspace{3em} $\hat{\ell}_t(i) = \begin{cases} 
\frac{n}{\epsilon} \ell_t(i_t), & \text{for } i = i_t \\
0, & \text{OW}
\end{cases}$
8: \hspace{1em} $x_{t+1} = \Pi_K(x_t - \eta \hat{\ell}_t)$
9: \hspace{1em} else
10: \hspace{2em} Play $i_t \sim x_t$
11: \hspace{2em} $\hat{\ell}_t = 0$, $x_{t+1} = x_t$. 

Simple MAB algorithm

- $\mathbb{E}[\hat{\ell}_t] = \ell_t$ and $\mathbb{E}[\hat{f}_t(x_t)] = \mathbb{E}[\hat{\ell}_t^\top x_t] = f_t(x_t)$

- Expected regret bound when $\epsilon = n^{2/3}T^{-1/3}$

$$
\mathbb{E}[\sum_{t=1}^{T} \ell_t(i_t)] - \min_i \sum_{t=1}^{T} \ell_t(i) \leq O(T^{2/3}n^{2/3})
$$
Combining exploration and exploitation steps (Auer et al. 2002b).

**Algorithm 2** EXP3 - simple version

1: Choose $\epsilon > 0, x_1 = [1/n, \ldots, 1/n]$.
2: **for** $t = 1, \ldots, T$ **do**
3: Choose $i_t \sim x_t$ and receive $l_t(i_t)$.
4: Let
   \[ \hat{l}_t(i) = \begin{cases} \frac{l_t(i_t)}{x_t(i_t)}, & \text{for } i = i_t \\ 0, & \text{OW} \end{cases} \]
5: Update $y_{t+1}(i) = x_t(i)e^{-\epsilon \hat{l}_t(i)}$, $x_{t+1} = \frac{y_{t+1}}{\|y_{t+1}\|_1}$

- $\mathbb{E}[\hat{l}_t] = l_t$
- Choose $\epsilon = \sqrt{\frac{\log n}{Tn}}$, expected regret bound $O(\sqrt{Tn\log n})$
Stochastic Multi-armed Bandit

Setting

- Player chooses $i_t \in \{1, \ldots, n\}$.
- Each action $i_t$ has a reward $r_{it}$ from a (fixed) probability distribution $\mathbb{P}_{it}$ with mean $\mu_{it}$.
- The reward revealed to the player is a sample taken from $\mathbb{P}_{it}$.

A sub case: Bernoulli Multi-armed Bandit with $\mathbb{P}_i = \text{Bernoulli}(p_i)$, $r_i \in \{0, 1\}$. 
Algorithm 3 Bernoulli Multi-armed Bandit

1: Set $N = Q = S = F = 0 \in \mathbb{R}^n$.
2: for $t = 1, \ldots, T$ do
3: \hspace{1em} $i_t = \text{PickArm}(Q, N, S, F)$
4: \hspace{1em} $r_t = \text{BernoulliReward}(i_t)$
5: \hspace{1em} $N[i_t] = N[i_t] + 1$ (number of times arm $i$ is pulled)
6: \hspace{1em} $Q[i_t] = Q[i_t] + \frac{(r_t - Q[i_t])}{N[i_t]}$ (empirical average reward of pulling $i$)
7: \hspace{1em} $S[i_t] = S[i_t] + r_t$ (number of times a reward of 1 was received)
8: \hspace{1em} $F[i_t] = F[i_t] + (1 - r_t)$ (number of times a reward of 0 was received)
Random selection
\( \epsilon \)-Greedy algorithm
Boltzmann Exploration
Upper Confidence Bounds
Bayesian UCB
Thompson Sampling

...
Using one sided Hoeffding’s inequality

\[ \mathbb{P}(\mu_i \geq Q[i] + \epsilon) \leq e^{-2N[i]\epsilon^2} \]

UCB strategy

\[ i = \arg\max_i (Q[i] + \epsilon), \text{ where } \epsilon = \sqrt{\frac{2\log(t)}{N[i]}} \]

Expected regret bound: \( O(\log(T)) \) (Auer et al. 2002a)
Thompson Sampling Strategy

**Beta distribution** $\text{Beta}(\alpha, \beta)$

$$f(x; \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1 - x)^{\beta-1}$$

**Thompson Sampling algorithm:**

- Initialize $p_i \sim \text{Beta}(1, 1), \forall i$

- **for** $t = 1, \ldots, T$

  $$Q[i] \sim \text{Beta}(S[i] + 1, F[i] + 1), \forall i$$

  $$i_t = \text{argmax}_i\{Q[i]\}$$

Expected regret bound: $O(\log(T))$ (Agrawal and Goyal 2012)

Generalize to $\tilde{r} \in [0, 1]$: after observing reward $\tilde{r}_t$, perform

$$r_t \sim \text{Bernoulli}(\tilde{r}_t)$$
**Thompson Sampling Strategy**

**Beta distribution** $\text{Beta}(\alpha, \beta)$

$$f(x; \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1} (1 - x)^{\beta-1}$$

**Thompson Sampling algorithm:**

- Initialize $p_i \sim \text{Beta}(1, 1), \forall i$
- for $t = 1, \ldots, T$

  $$Q[i] \sim \text{Beta}(S[i] + 1, F[i] + 1), \forall i$$

  $$i_t = \arg\max_i \{Q[i]\}$$

**Expected regret bound:** $O(\log(T))$ (Agrawal and Goyal 2012)

**Generalize to $\tilde{r} \in [0, 1]$:** after observing reward $\tilde{r}_t$, perform

$$r_t \sim \text{BernoulliReward}(\tilde{r}_t)$$


