Adaptive Methods for Optimization in Stochastic Environments

Xuedong Shang (xuedong.shang@inria.fr)
Supervised by **Emilie Kaufmann** and **Michal Valko**
Inria Lille, SequeL Team

February 2, 2018

CRiSTAL & Inria Lille

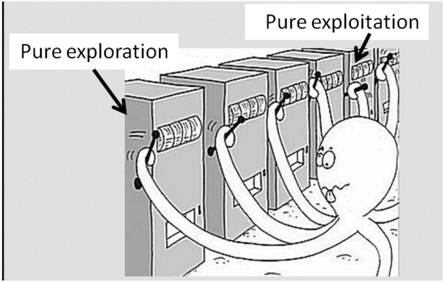
Multi-armed Bandit

Multi-armed Bandit Game

- What is a MAB game?

Multi-armed Bandit Game

- What is a MAB game?
- Objective: maximize the total reward



Source: Microsoft Research

More Formally...

- **What we know:** the number of arms K
- **What we do not know:** the reward distributions $(\nu_k)_{1 \leq k \leq K}$ of the arms

More Formally...

- **What we know:** the number of arms K
- **What we do not know:** the reward distributions $(\nu_k)_{1\leq k\leq K}$ of the arms
- At time t, a player pulls one arm $k_t \in \{1, \ldots, K\}$ and receives a reward $r_t \sim \nu_{k_t}$

More Formally...

- **What we know:** the number of arms K
- **What we do not know:** the reward distributions $(\nu_k)_{1 \leq k \leq K}$ of the arms
- At time t, a player pulls one arm $k_t \in \{1, \ldots, K\}$ and receives a reward $r_t \sim \nu_{k_t}$
- A *policy* chooses one arm k_t to play

More Formally...

- **What we know:** the number of arms K
- **What we do not know:** the reward distributions $(\nu_k)_{1 \leq k \leq K}$ of the arms
- At time t, a player pulls one arm $k_t \in \{1, \ldots, K\}$ and receives a reward $r_t \sim \nu_{k_t}$
- A *policy* chooses one arm k_t to play
- We want to minimize the (cumulative) regret:

$$R_n = n\mu^* - \sum_{1 \leq t \leq n} \mu_{k_t}$$

More Formally...

- **What we know:** the number of arms K
- **What we do not know:** the reward distributions $(\nu_k)_{1 \leq k \leq K}$ of the arms
- At time t, a player pulls one arm $k_t \in \{1, \ldots, K\}$ and receives a reward $r_t \sim \nu_{k_t}$
- A *policy* chooses one arm k_t to play
- We want to minimize the (cumulative) regret:

$$R_n = n\mu^* - \sum_{1 \leq t \leq n} \mu_{k_t}$$

- However, this does not seem to be always the right way to base the strategies on in some scenarios...

Fixed-budget Pure Exploration Game

- **What we know:** the budget n, the number of arms K
- **What we do not know:** the reward distributions $(\nu_k)_{1 \leq k \leq K}$ of the arms

Fixed-budget Pure Exploration Game

- **What we know:** the budget n, the number of arms K
- **What we do not know:** the reward distributions $(\nu_k)_{1 \leq k \leq K}$ of the arms
- For each round $t = 1, 2, \cdots, n$, a player pulls one arm and receives a reward as before, but this time, the policy will output a recommendation j_n at the end of n rounds

Fixed-budget Pure Exploration Game

- **What we know:** the budget n, the number of arms K
- **What we do not know:** the reward distributions $(\nu_k)_{1 \leq k \leq K}$ of the arms
- For each round $t = 1, 2, \cdots, n$, a player pulls one arm and receives a reward as before, but this time, the policy will output a recommendation j_n at the end of n rounds
- We want to minimize the (simple) regret:

$$S_n = \mu^* - \mu_{j_n}$$

Fixed-budget Pure Exploration Game

- **What we know:** the budget n, the number of arms K
- **What we do not know:** the reward distributions $(\nu_k)_{1 \leq k \leq K}$ of the arms
- For each round $t = 1, 2, \cdots, n$, a player pulls one arm and receives a reward as before, but this time, the policy will output a recommendation j_n at the end of n rounds
- We want to minimize the (simple) regret:

$$S_n = \mu^* - \mu_{j_n}$$

Current State-of-the-Art

- Finitely-armed algorithms: Successive Reject [Audibert et al. 2010], Sequential Halving [Karnin et al. 2013], UGapE [Gabillon et al. 2013]...
- Infinitely-armed algorithms: SiRI [Carpentier and Valko 2015], Hyperband [Li et al. 2017]

Black Box Optimization and Beyond...

Reformulation in the context of Optimization

- An unknown noisy function $f : \mathcal{X} \to \mathbb{R}$.
- At each step t, a policy picks an action $\mathsf{x_t} \in \mathcal{X}$ and receives a reward $r_t = f(\mathsf{x_t}) + \epsilon_\mathsf{t}$ where ϵ_t is the noise.
- **Simple regret:**

$$S_n = f(\mathsf{x}^*) - f(\mathsf{x}_{j_n}).$$

- **Cumulative regret:**

$$R_n = \sum_{1 \leq t \leq n} (f(\mathsf{x}^*) - f(\mathsf{x}_t)).$$

Current State-of-the-Art

- **Hierarchical Optimization**: HOO [Bubeck et al. 2011], POO [Grill et al. 2015], HCT Gheshlaghi-Azar et al. 2014]...
- **Bayesian Optimization**: GP-UCB [Srinivas et al. 2009], TPE [Bergstra et al. 2011]...

Current State-of-the-Art

- **Hierarchical Optimization**: HOO [Bubeck et al. 2011], POO [Grill et al. 2015], HCT Gheshlaghi-Azar et al. 2014]...
- **Bayesian Optimization**: GP-UCB [Srinivas et al. 2009], TPE [Bergstra et al. 2011]...
- **Perspective**:
    - New best arm identification algorithms based on hierarchical exploration?
    - Adaptive partitioning?
    - Anytime?

Hyperparameter Optimization

Experiment

- select a set of hyper-parameters $\mathbf{x}_t$ as an arm
- $\mathbf{x}_t$ is then used in some machine learning classifier
- recommend an arm $\mathbf{x}_{j_t}$

- **Loss function**:
  - Logistic loss for classification problems
  - Mean squared error for regression problems
- The underlying task is to find some classifier $g_{\mathbf{x}_t}$ which minimizes the expected loss $f(g_{\mathbf{x}_t}) = \mathbb{E}\left[\mathcal{L}(\boldsymbol{y}, g_{\mathbf{x}_t}(X)\right]$

Hyperband

- Hyperband consists of several brackets/iterations

Hyperband

- Hyperband consists of several brackets/iterations
- At each bracket i, given B and N_i (budget can be time, epochs, dataset subsampling, etc):
  - sample randomly N_i configurations
  - run Sequential Halving based on validation losses

Hyperband

- Hyperband consists of several brackets/iterations
- At each bracket i, given B and N_i (budget can be time, epochs, dataset subsampling, etc):
  - sample randomly N_i configurations
  - run Sequential Halving based on validation losses
- Trade-off between N_i and B/N_i

Hyperband

- Hyperband consists of several brackets/iterations
- At each bracket i, given B and N_i (budget can be time, epochs, dataset subsampling, etc):
    - sample randomly N_i configurations
    - run Sequential Halving based on validation losses
- Trade-off between N_i and B/N_i
- **Output:** the best intermediate loss ever seen

Beyond Hyperband?

- Pros: strong anytime performance, easily parallelizable
- Cons: convergence to global optimum heavily limited by its reliance on randomly-drawn configurations

Beyond Hyperband?

- Pros: strong anytime performance, easily parallelizable
- Cons: convergence to global optimum heavily limited by its reliance on randomly-drawn configurations
- **Perspective:** take into account previously sampled configurations? $\rightarrow$ TPE+Hyperband [Falkner et al. 2017]

Contextual Bandits and Algorithm Selection

Contextual Bandits

- At time t:
    - receive some context $c_t \in C$
    - the player pulls an arm $k_t \in \{1, \ldots, K\}$
    - receive some reward $r_t(k_t)$

Contextual Bandits

- At time t:
  - receive some context $c_t \in C$
  - the player pulls an arm $k_t \in \{1, \ldots, K\}$
  - receive some reward $r_t(k_t)$
- A policy $\pi \in \Pi = \{\pi : C \rightarrow 1 \cdots K\}$ chooses one arm k_t to play

Contextual Bandits

- At time t:
  - receive some context $c_t \in C$
  - the player pulls an arm $k_t \in \{1, \ldots, K\}$
  - receive some reward $r_t(k_t)$
- A policy $\pi \in \Pi = \{\pi : C \to 1 \cdots K\}$ chooses one arm k_t to play
- We want to minimize the (cumulative) regret:

$$R_n = \max_{\pi \in \Pi} \sum_{1 \leq t \leq n} r_t(\pi(c_t)) - \sum_{1 \leq t \leq n} r_t(k_t)$$

Online Algorithm Selection

- **Idea:** a set of complementary algorithms performing well on different instances

Online Algorithm Selection

- **Idea:** a set of complementary algorithms performing well on different instances
- Using supervised learning techniques to build a selection mapping $\lambda : \mathrm{instance} \rightarrow \mathrm{algorithm}$
- Each instance is characterized by a set of features
- Online setting: initialize λ with offline training data, then make predictions for online new instances

Online Algorithm Selection

- **Idea:** a set of complementary algorithms performing well on different instances
- Using supervised learning techniques to build a selection mapping $\lambda : \text{instance} \rightarrow \text{algorithm}$
- Each instance is characterized by a set of features
- Online setting: initialize λ with offline training data, then make predictions for online new instances
- Can be seen as a contextual bandit problem
- **Perspective:** LinUCB [Li et al. 2010]? Comparable to greedy approach?

Thank you!