MaxHedge: Maximizing a Maximum Online

Stephen Pasteris (UCL), **Fabio Vitale** (Sapienza Unversity, Rome),
Kevin Chan (ARL), **Shiqiang Wang** (IBM US), **Mark Herbster** (UCL)

Overview

Learning proceeds in trials: on every trial the learner selects a subset of possible servers on which to place a service, subject to a budget. For each server, selecting the server incurs a cost. After selecting the servers a user requires the use of the services, paying us a reward based on how far away the *nearest* selected server is. The objective is to maximize the cumulative profit.

Technical challenge: The *maximum reward* among all placed services is received by the system, which makes the problem different from classic online learning problems that usually consider the sum reward.

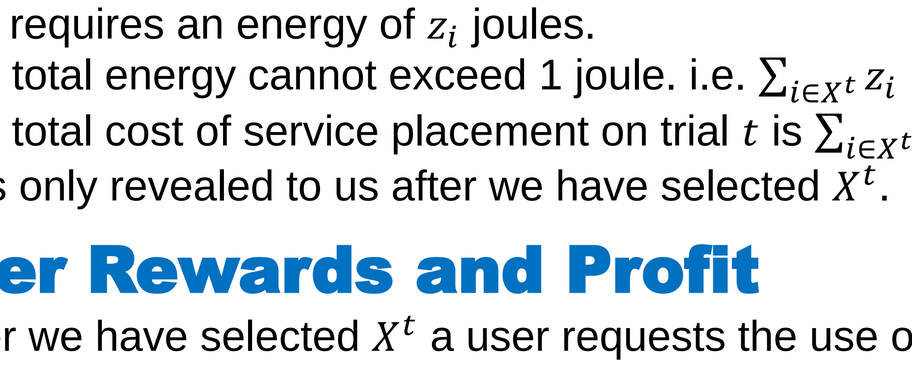
Military relevance: The location of the next analytics request is usually unknown beforehand, especially when crossing coalition boundaries. Our approach learns the likely locations of future requests over time and places services accordingly.

Placing the Service

- We have a set of n servers.
- Learning proceeds in trials $t = 1,2,3 \ldots, T$.
- On every trial t we select a set $X^t \subseteq [n]$ of servers on which to place the service.
- Placing the service on server $i \in [n]$ on trial t costs us £c_i^t and requires an energy of z_i joules.
- The total energy cannot exceed 1 joule. i.e. $\sum_{i \in X^t} z_i \leq 1$.
- The total cost of service placement on trial t is $\sum_{i \in X^t} c_i^t$.
- c^t is only revealed to us after we have selected X^t.

User Rewards and Profit

- After we have selected X^t a user requests the use of the service.
- If the nearest server (to the user) in X^t is server i then the user rewards us with r_i^t.
- r^t is only revealed to us after we have selected X^t.
- The profit at trial t is defined as the reward minus the cost: $\mu^t(X^t) = \max_i r_i^t - \sum_{i \in X^t} c_i^t$
- The goal is to maximize the cumulative profit: $\sum_{t \in [T]} \mu^t(X^t)$



Special Cases

The following special cases of our problem are online learning versions of classic computer science problems:

- (Inverted) Facility Location Problem: $z_i = 0, c_i^t \geq 0$
- (Inverted) Knapsack Median Problem: $c_i^t = 0$
- 0-1 Knapsack Problem: $r_i^t = 0, c_i^t \leq 0$

In this poster we only consider the cases in which $c_i^t \geq 0$.

Performance of MaxHedge

- For some $\alpha, \delta \in [0,1]$, some subset $S \subseteq [n]$, and some trial t define:
$$\hat{\mu}_{\alpha,\delta}^t(S) = \alpha \max_{i \in S} r_i^t - \delta \sum_{i \in S: c_i^t > 0} c_i^t$$
which would be the profit obtained on trial t if the placed the service on the servers in S, all the rewards were multiplied by α and all costs were multiplied by δ.
- We define: $\beta = \max_{i \in S} z_i,\ \delta = (1 - \sqrt{\beta})^2, \alpha = 1 - \exp(-\delta)$
$$\hat{r} = \max_{t \in [T], i \in [n]} r_i^t,\ \hat{c} = \max_{t \in [T], i \in [n]} c_i^t$$
- For any $S \subseteq [n]$ with $\sum_{i \in S} z_i \leq 1$, the total cumulative profit of MaxHedge is at least:
$$\sum_{t \in [T]} \hat{\mu}_{\alpha,\delta}^t(S) - n\delta(\hat{r} + \hat{c})\sqrt{(2T)}$$
- MaxHedge takes a time of only $O(n \log n)$ per trial.

How MaxHedge Works

- We define:
$$\mathrm{C} = \{x \in [0,1]^n : \sum_{i \in n} z_i x_i \leq 1$$
which is a relaxation of the set of feasible service placements.
- A vector $\omega \in C$ is maintained throughout. Let ω^t be value of ω at the start of trial t.
- X^t is constructed from ω^t in a randomized way.
- After c^t and r^t are received a convex function h^t is constructed in which $h^t(\omega^t)$ is lower bound on the negative of the expected profit via the above randomized construction.
- The gradient g, of h^t at ω^t is computed.
- ω^{t+1} is then the Euclidean projection of $\omega^t + \eta g$ onto C, for learning rate η.

Algorithm 1 Constructing X^t

$C \leftarrow \{\boldsymbol{x} \in [0,1]^n : \langle \boldsymbol{x}, \boldsymbol{z} \rangle \leq 1\}$
$\beta \leftarrow \max_{i \in [n]} z_i$
$\tau \leftarrow 1 - \sqrt{\beta}$
$\delta \leftarrow \left(1 - \sqrt{\beta}\right)^2$.
$\Gamma \leftarrow \{q \in \mathbb{N} : \exists i \in [n] \text{ with } \tau^q \beta < z_i \leq \tau^{q-1}\beta\}$
For all $q \in \Gamma$ set $\Omega_q \leftarrow \{i \in [n] : \tau^q \beta < z_i \leq \tau^{q-1}\beta\}$.
Input: $\boldsymbol{\omega}^t \in C$
For all $q \in \Gamma$ set $\pi_q^t \leftarrow \sum_{i \in \Omega_q} \omega_i^t$
For all $q \in \Gamma$ set $\zeta_q^t \leftarrow \lfloor \delta \pi_q^t \rfloor$
For all $q \in \Gamma$ and for all $k \leq \zeta_q^t$ draw $\xi_{q,k}^t$ randomly from Ω_q such that $\xi_{q,k}^t \leftarrow i$ with probability ω_i^t / π_q^t
For all $q \in \Gamma$ and for $k := \zeta_q^t + 1$ draw $\xi_{q,k}^t$ randomly from $\Omega_q \cup \{0\}$ such that, for $i \in \Omega_q$ we have $\xi_{q,k}^t \leftarrow i$ with probability $(\delta \pi_q^t - \lfloor \delta \pi_q^t \rfloor)\omega_i^t / \pi_q^t$, and we have $\xi_{q,k}^t \leftarrow 0$ with probability $\lfloor \delta \pi_q^t \rfloor + 1 - \delta \pi_q^t$. NB: In the case that $\pi_q^t = 0$ we define $0/0 = 0$
Output: $X^t \leftarrow \{\xi_{q,k}^t : q \in \Gamma, k \leq \zeta_q^t + 1\} \setminus \{0\}$

Algorithm 2 Computing $\boldsymbol{\omega}^{t+1}$

$C \leftarrow \{\boldsymbol{x} \in [0,1]^n : \langle \boldsymbol{x}, \boldsymbol{z} \rangle \leq 1\}$
$\beta \leftarrow \max_{i \in [n]} z_i$
$\delta \leftarrow \left(1 - \sqrt{\beta}\right)^2$
Input: $\boldsymbol{\omega}^t \in C$, $\boldsymbol{c}^t \in \mathbb{R}^n$ $\boldsymbol{r}^t \in \mathcal{P}$
Order $[n]$ as $[n] = \{\sigma(t,1), \sigma(t,2), \ldots \sigma(t,n)\}$ such that $r_{\sigma(t,j)}^t \geq r_{\sigma(t,j+1)}^t$ for all $j \in [n-1]$
For all $j \in [n]$ set $\epsilon_j^t \leftarrow \exp\left(-\delta \sum_{k=1}^j \omega_{\sigma(t,k)}^t\right)$
For all $j \in [n]$ set:
$\lambda_j^t \leftarrow r_{\sigma(t,n)}^t \epsilon_n^t + \sum_{k=j}^{n-1} \left(r_{\sigma(t,k)}^t - r_{\sigma(t,k+1)}^t\right) \epsilon_k^t$
For all $j \in [n]$ set:
$g_{\sigma(t,j)}^t \leftarrow \delta \left(c_{\sigma(t,j)}^{t+} + c_{\sigma(t,j)}^{t-} \exp\left(-\delta \omega_{\sigma(t,j)}^t\right) - \lambda_j^t\right)$
$\hat{\eta}^t \leftarrow \min\left\{\hat{\eta}^{t-1}, \sqrt{n}/\|\boldsymbol{g}^t\|\right\}$
$\eta^t \leftarrow \hat{\eta}^t / \sqrt{2t}$
$\boldsymbol{y}^t \leftarrow \boldsymbol{\omega}^t - \eta^t \boldsymbol{g}^t$
Output: $\boldsymbol{\omega}^{t+1} \leftarrow \Pi_C(\boldsymbol{y}^t)$

Full paper of this work: S. Pasteris, F. Vitale, K. Chan, S. Wang, M. Herbster, "MaxHedge: Maximising a Maximum Online," in AISTATS, Apr. 2019.
Related paper: S. Pasteris, S. Wang, M. Herbster, T. He, "Service placement with provable guarantees in heterogeneous edge computing systems," in IEEE INFOCOM, Apr. 2019.

Distributed Analytics and Information Science International Technology Alliance
Annual Fall Meeting, September 2019