Adversarial Online Learning

Naram Mhaisen

TU Delft

The General Model of an Online System

Across each time slot $t = 1,2,\ldots T$:

- **The Learner** takes action. $x_t \in \mathcal{X}$
- **The Env.** reveals a cost signal. $f_t(x_t, s_t) : \mathcal{X} \to \mathbb{R}$
- **The Env.** transitions to some new state s_{t+1}

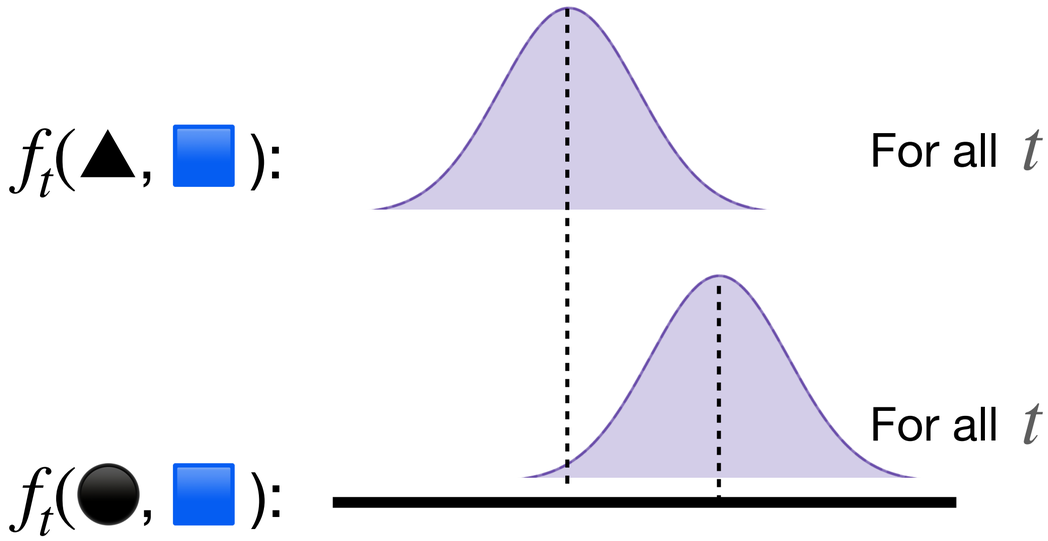
f_t is unknown to the learner before committing to x_t

The Model of an Online System - RL

Across each time slot $t = 1,2,\ldots T$:

- **The Learner** takes action. $x_t \in \mathcal{X}$
- **The Env.** reveals a cost signal. $f_t(x_t, s_t) : \mathcal{X} \to \mathbb{R}$
- **The Env.** transitions to some new state s_{t+1}

- f_t is unknown to the learner before committing to x_t.
- **RL:** A fixed dist. for each (x_t, s_t) pair:



The Model of an Online System - RL

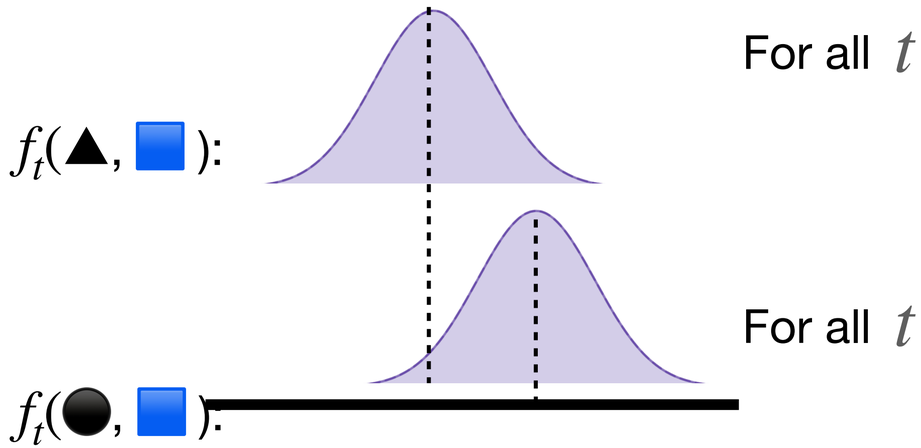
Across each time slot $t = 1,2,\ldots T$:

- **The Learner** takes action.
  $x_t \in \mathcal{X}$
- **The Env.** reveals a cost signal.
  $f_t(x_t, s_t) : \mathcal{X} \to \mathbb{R}$
- **The Env.** transitions to some new state s_{t+1}

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

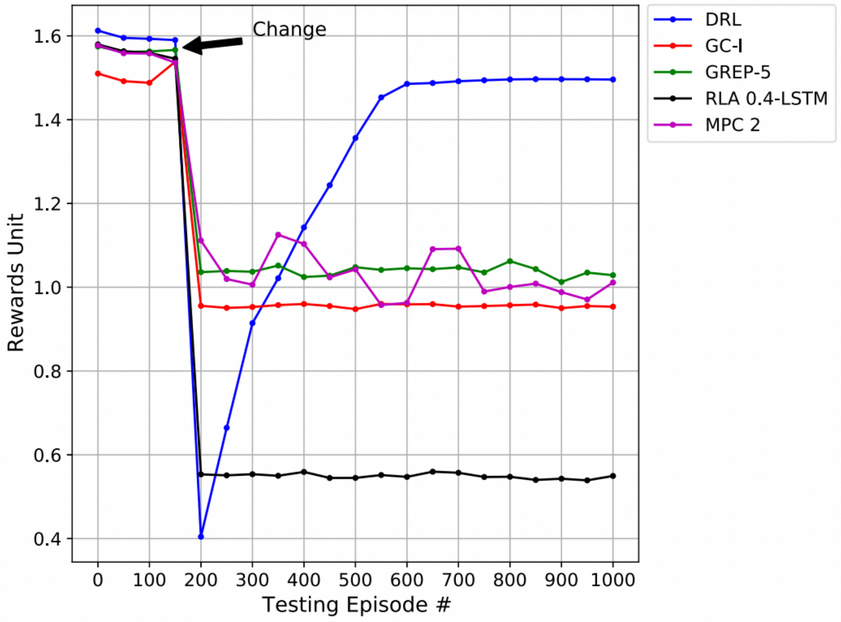
Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal



Detour

But RL agents can already handle change?

On Designing Smart Agents for Service Provisioning in Blockchain-Powered Systems



- Early changes
- Very few
- Still can experience performance loss
- Requires discount rate tuning

How about an adversarial change *at every* time slot?

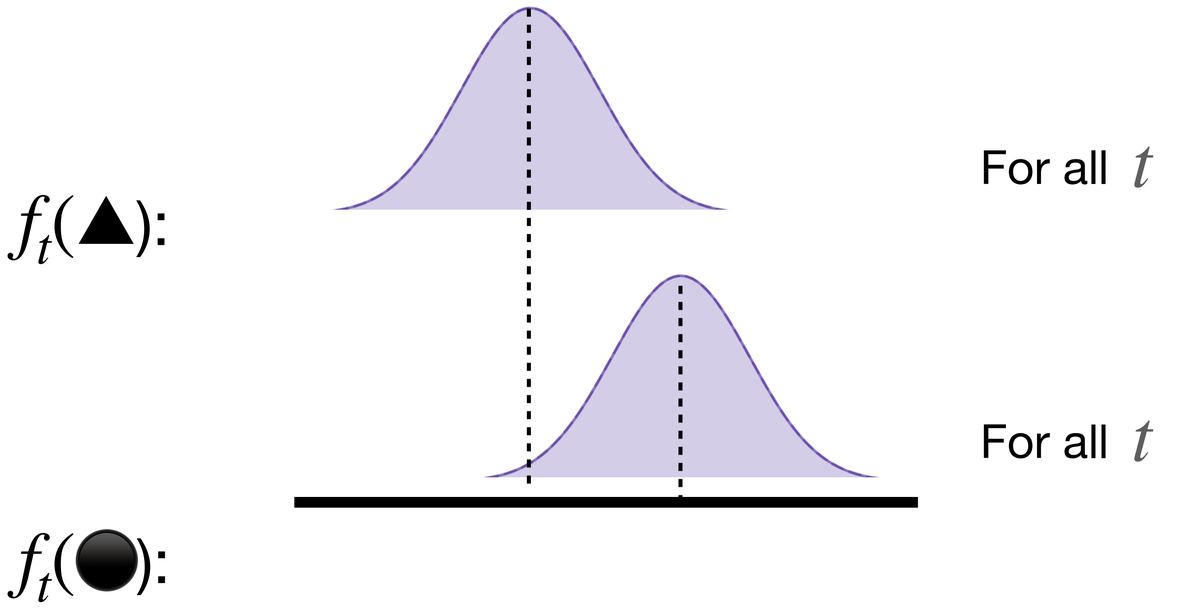
The Model of an Online *Stateless* System

a.k.a "Bandits"

Across each time slot $t = 1,2,\ldots T$:

- **The Learner** takes action. $x_t \in \mathcal{X}$
- **The Env.** reveals a cost signal. $f_t(x_t) : \mathcal{X} \to \mathbb{R}$
- ~~**The Env.** transitions to some new state s_{t+1}~~

- f_t is unknown to the learner before committing to x_t.
- Bandits (A fixed dist. for each (x_t)):



The Model of an Online *Adversarial* *Stateless* System

a.k.a "Adversarial Bandits"

Across each time slot $t = 1,2,\ldots T$:

- **The Learner** takes action. $x_t \in \mathcal{X}$
- **The Env.** reveals a cost signal. $f_t(x_t) : \mathcal{X} \to \mathbb{R}$
- ~~**The Env.** transitions to some new state s_{t+1}~~

- f_t is unknown to the learner before committing to x_t.
- ***f_t can be any convex function that changes every t.***

The Model of an Online *Adversarial* *Stateless* System with *Full Feedback*

a.k.a "Online Convex Optimization (OCO)" Framework

Across each time slot $t = 1,2,\ldots T$:

- **The Learner** takes action. $x_t \in \mathcal{X}$
- **The Env.** reveals a cost function. $f_t(\cdot) : \mathcal{X} \to \mathbb{R}$
- **The Learner** "suffers" action. $f_t(x_t)$
- ~~**The Env.** transitions to some new state s_{t+1}~~

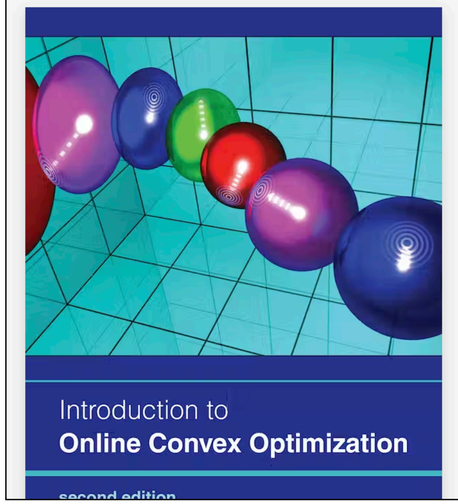
- f_t is unknown to the learner before committing to x_t.
- ***f_t can be any convex function that changes every t.***

The Model of an Online *Adversarial* *Stateless* System with *Full Feedback*

a.k.a “Online Convex Optimization (OCO)” Framework

Across each time slot $t = 1,2,\ldots T$:

- **The Learner** takes action. $x_t \in \mathcal{X}$
- **The Env.** reveals a cost function. $f_t(\cdot) : \mathcal{X} \to \mathbb{R}$
- **The Learner** “suffers” action. $f_t(x_t)$



From: Adaptive Computation and Machine Learning series

Introduction to Online Convex Optimization, Second Edition

By Elad Hazan

248 pp., 6 x 9 in, 11
Hardcover
ISBN: 9780262046985
Published: September 6, 2022

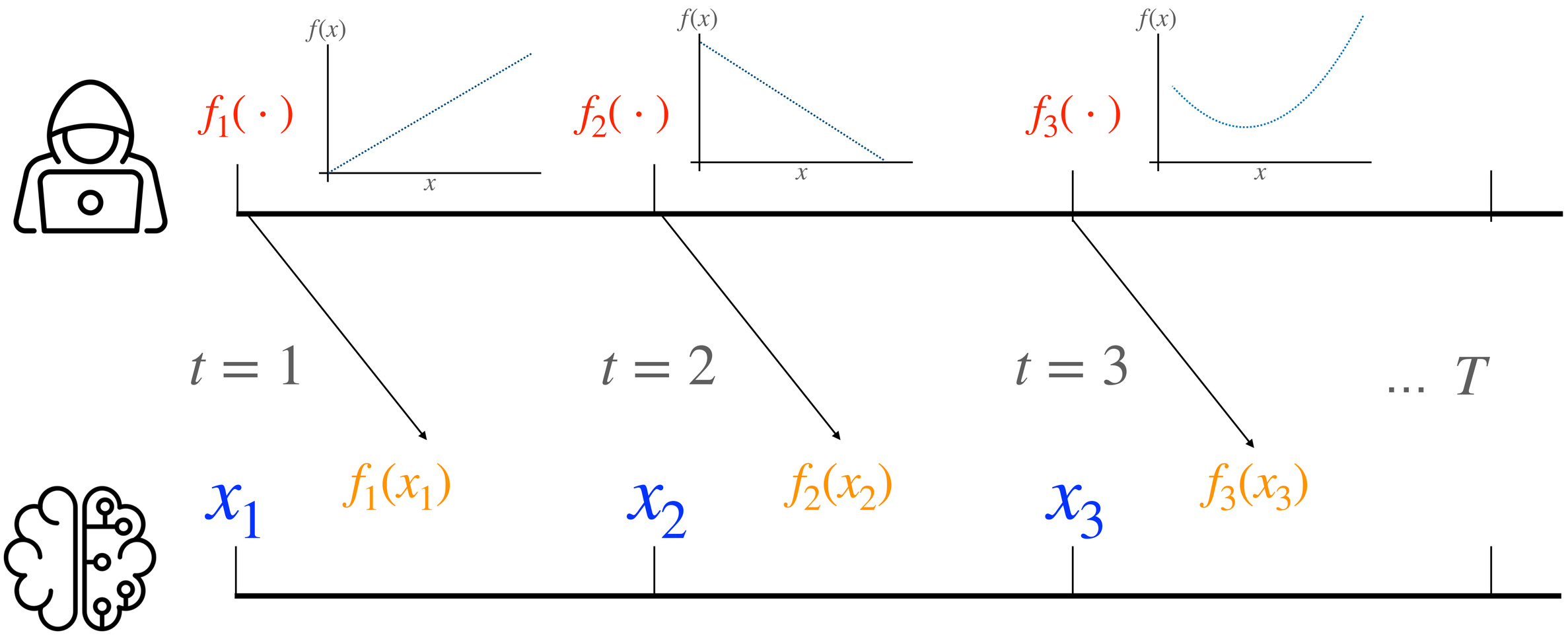
A Modern Introduction to Online Learning

Francesco Orabona
Boston University
francesco@orabona.com

March 22, 2022

The Model of an Online *Adversarial* *Stateless* System with *Full Feedback*

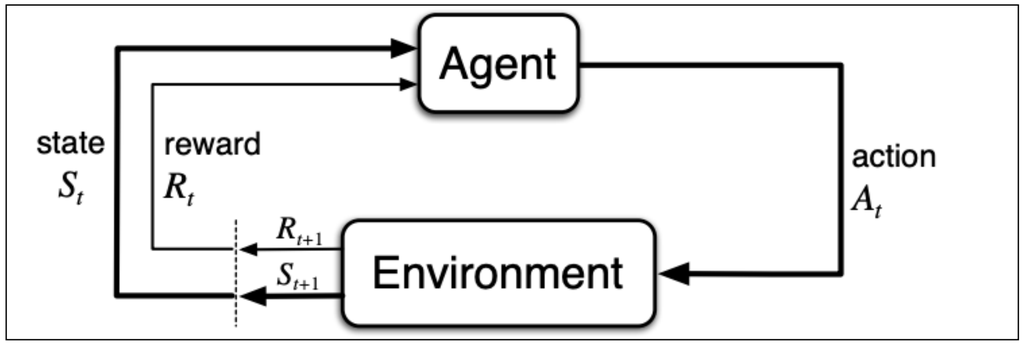
OCO (again) — Visually



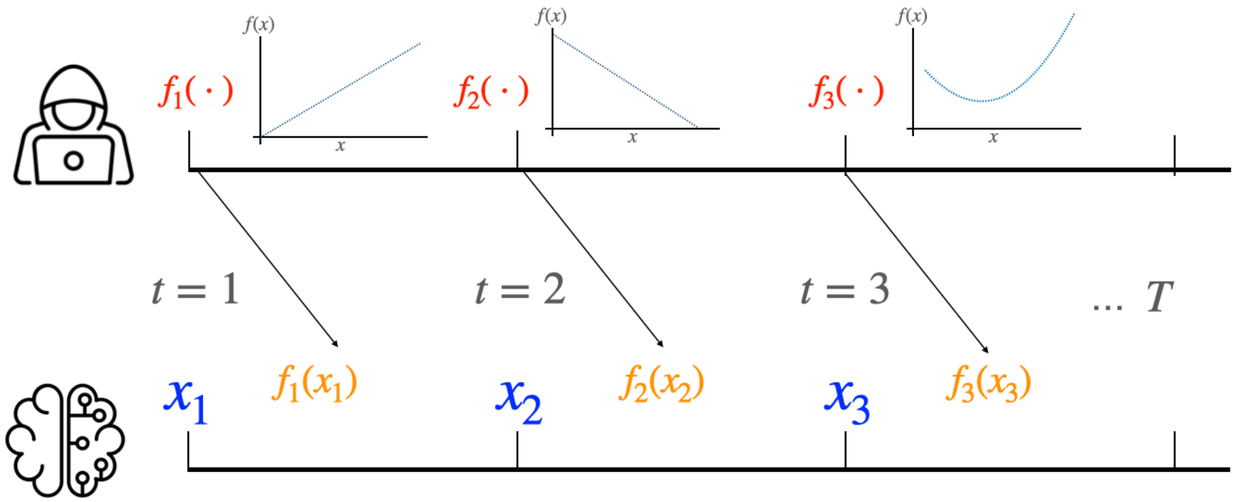
Check-point

Online Learning: learn from previous costs to minimize future unknown costs.

- **Reinforcement Learning:** Costs are samples from stationary unknown distributions/functions.
  - **Bandits Learning:** Costs are independent from the “state”.
- **Adversarial Learning:** Costs are generated through unknown and changing convex functions.
  - **OCO:** Costs are independent of the “state”.



Why is OCO important?



- There are important applications that are not stochastic.
  - Rapidly changing users and/or environment
- Subsumes the stochastic environment model.
  - Algorithm A “works” in OCO => “works” in a stochastic environment.

OCO Background (1/3)

Metric: “Regret”

- Given all the adversarial functions $f_1(\cdot), f_2(\cdot), \ldots f_T(\cdot)$ in advance, we can calculate and execute:

$$x^\star = \arg\min_{x \in \mathcal{X}} \sum_{t=1}^{T} f_t(x) \qquad \text{i.e.,:} \qquad x^\star = \arg\min_{x \in \mathcal{X}} f_{1:t}(x)$$

- An online alg. $\mathscr{A}(t-1)$ which outputs x_t based on info up until $t-1$.
  - Performance gap each step: $f_t\left(\mathscr{A}(t-1)\right) - f_t\left(x^\star\right)$

$$R_T = \sum_{t=1}^{T} f_t\left(\left(\mathscr{A}(t-1)\right) - f_t(x^\star)\right)$$

OCO Background (2/3)

“No regret”, “Achievability”, and “Linearization”

- An algorithm whose regret grows sub-linearly is called *“No-regret” Alg*.
  - If $R_T(\mathscr{A}) = O\left(T^{\alpha}\right)$ for some $\alpha < 1$, then the average regret goes to 0: $\frac{R_T}{T} \to 0$ as $T \to \infty$.
  - i.e., as time goes on, our performance gap will go to 0.
  - No algorithm can *achieve* better regret than $R_T(\mathscr{A}) = O\left(T^{0.5}\right)$. (i.e., $R_T(\mathscr{A}) = \Omega\left(T^{0.5}\right)$
- The regret for any sequence of functions is reducible to the regret of linear functions

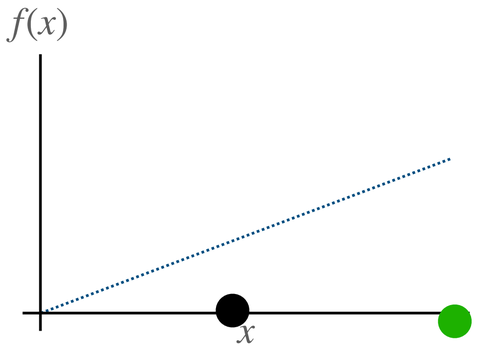
$$R_T = \sum_{t=1}^{T} \left(f_t(x_t) - f_t(x^\star)\right) \leq \sum_{t=1}^{T} \left(\langle c_t, x_t - x^\star \rangle\right), \qquad c_t = \nabla f_t(x_t)$$

OCO Background (3/3)

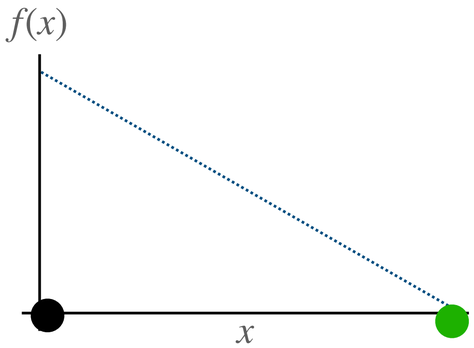
Solution Algorithms

- Following the “leader”:

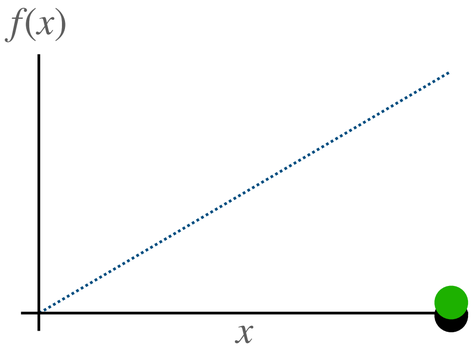
$$x_{t+1} = \arg\min_{x \in \mathcal{X}} \langle c_{1:t}, x \rangle$$



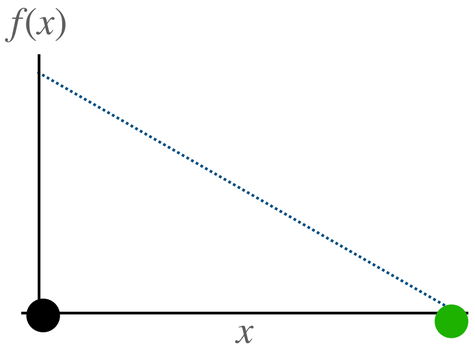
$x_1 = 0.5$



$x_2 = 0$



$x_3 = 1$



$x_4 = 0$

$x^\star = 1$ $\qquad$ $R_T \approx T/2$

OCO Background (3/3)

Solution Algorithms

- Add regularization: Following the regularized “leader” (FTRL):

$$x_{t+1} = \arg\min_{x \in \mathcal{X}} \left\{ \langle c_{1:t}, x \rangle + r_{1:t}(x) \right\}$$

- Example: $r_0(x) = \frac{1}{2\eta}\|x\|^2$, $r_t = 0,\ t \geq 2$

  - => $x_{t+1} = x_t + \eta c_t$ ← **Online Gradient Aascent**

Result: For any sequence of functions, FTRL with $\eta = O\left(1/\sqrt{T}\right)$

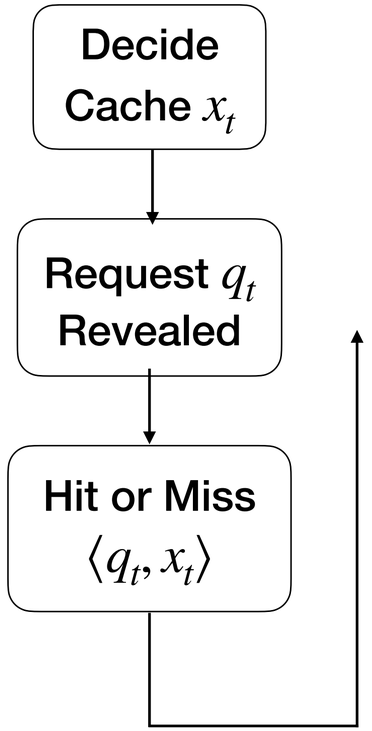
attains an optimal $R_T = O\left(\sqrt{T}\right)$

Use Case: Caching (1/2)

- A library of Equal unit size N files, Cache with Capacity C.
- At each time step, a request vector is revealed $q_t \in \{0,1\}^N$.
- The best caching configuration $x^\star$ is only known in hindsight via solving.

$$\text{maximize } \langle q_{1:T}, x \rangle$$

$$\text{s.t } x \in [0,1]^N \;\&\; \sum_{i=1}^{N} x_t^i \leq C$$



- If $q_t \sim$ Gaussian, then $x_{t+1} = \arg\max_{x \in \mathcal{X}} \langle q_{1:t}, x \rangle$, will yield iterates

$$\|x_t - x^\star\| \propto e^{-t} \quad (R_T \propto \log(T))$$

- OCO allows us to drop all assumptions on the process generating q_t.

Use Case: Caching (2/2)

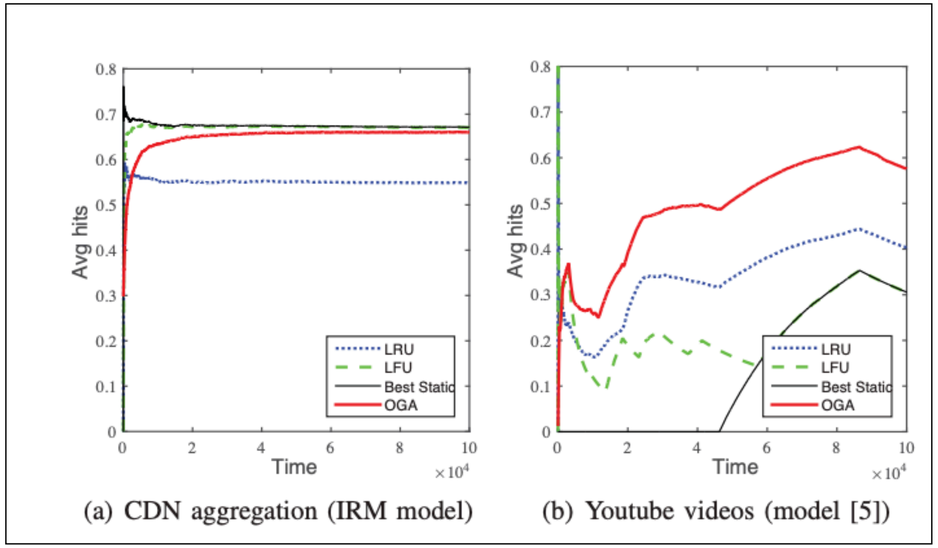
- The utility function: $f_t(x_t) = \langle x_t, q_t \rangle$
  - Convex function
  - Fully parametrized by q_t (Full information)
- Using the update rule:

$$x_{t+1} = \Pi_{\mathcal{X}} \left(x_t + \eta q_t \right)$$

- $\|x_t - x^*\| \propto 1/\sqrt{T}$ and $R_T = O\left(\sqrt{T}\right)$

Learning to Cache With No Regrets

Georgios S. Paschos*, Apostolos Destounis*, Luigi Vigneri*†, George Iosifidis‡
*France Research Center, Huawei Technologies, Paris
†IOTA Foundation, Berlin, Germany
‡School of Computer Science and Statistics, Trinity College Dublin, Ireland



(a) CDN aggregation (IRM model) (b) Youtube videos (model [5])

"Learning to Cache With No Regrets", *infocom'19*
"Online Caching Networks with Adversarial Guarantees", *sigmertrics'21*
"LeadCache: Regret-Optimal Caching in Networks", *neurips'21*

Optimistic OCO

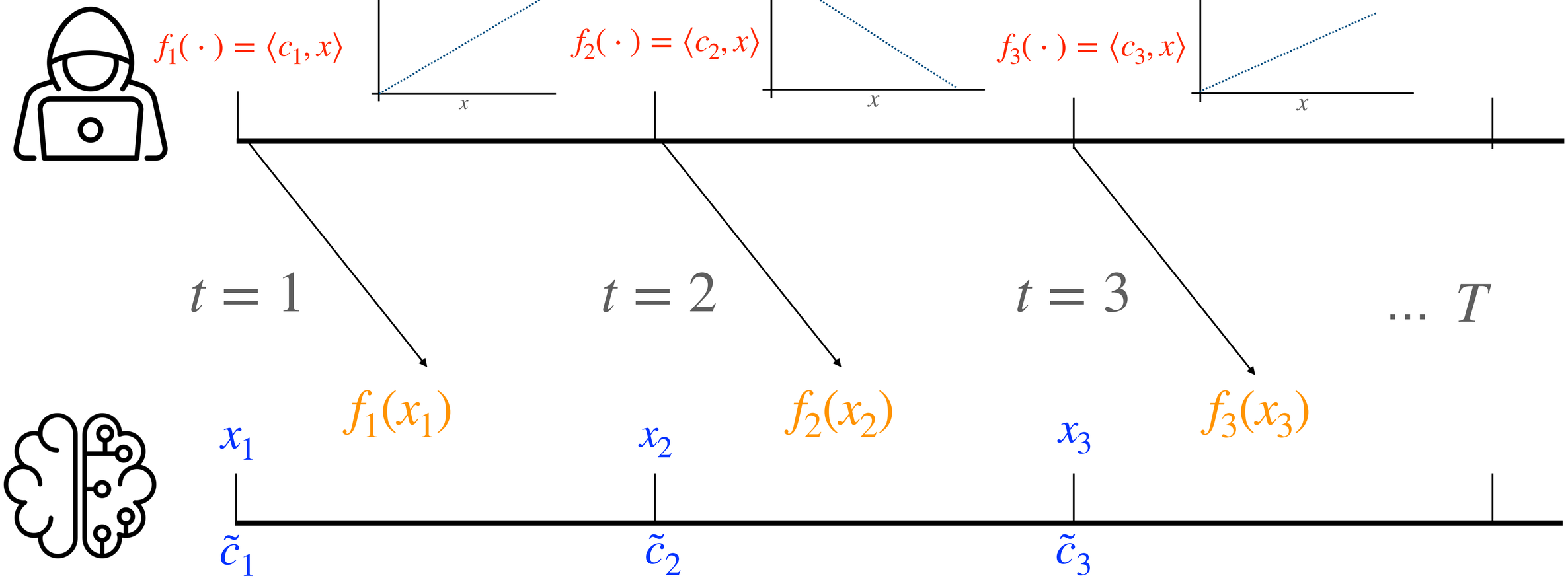
How to Incorporate “predictions”

- In RL & bandits: The future is likely to look similar to the past.
- In OCO: No guarantees about future cost.
  - Still, sometimes a plausible prediction is possible
  - No assumption on the quality of predictions is allowed

Can we leverage predictions *of unknown quality* to minimize the regret beyond the $\sqrt{T}$ limit *while still maintaining $\sqrt{T}$ even when all predictions fail?*

Optimistic OCO

Visually



$c_t \quad \tilde{c}_t$ Are completely independent

Optimistic OCO

OFTRL & Guarantees

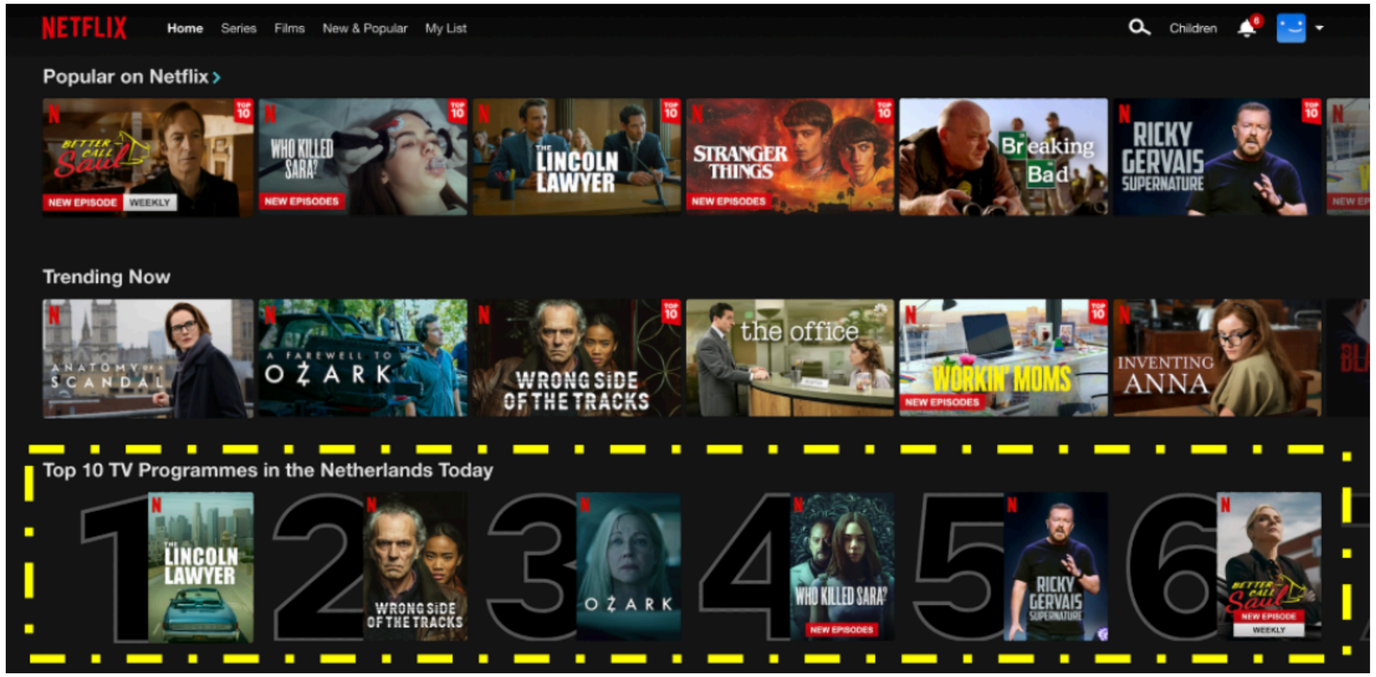
- Update step: $x_{t+1} = \arg\min_{x \in \mathcal{X}} \left\{ \langle c_{1:t} + \tilde{c}_{t+1}, x \rangle + r_{1:t}(x) \right\}$
- Regularization: $r_t(x) \propto \sum_{i=1}^{t-1} \|c_i - \tilde{c}_i\| \times \|x\|$
  - Perfect predictions: $x_{t+1} = \arg\min_{x \in \mathcal{X}} \left\{ \langle c_{1:t} + c_{t+1}, x \rangle \right\}$
  - Otherwise, stable iterates
  - A safe way to incorporate predictions

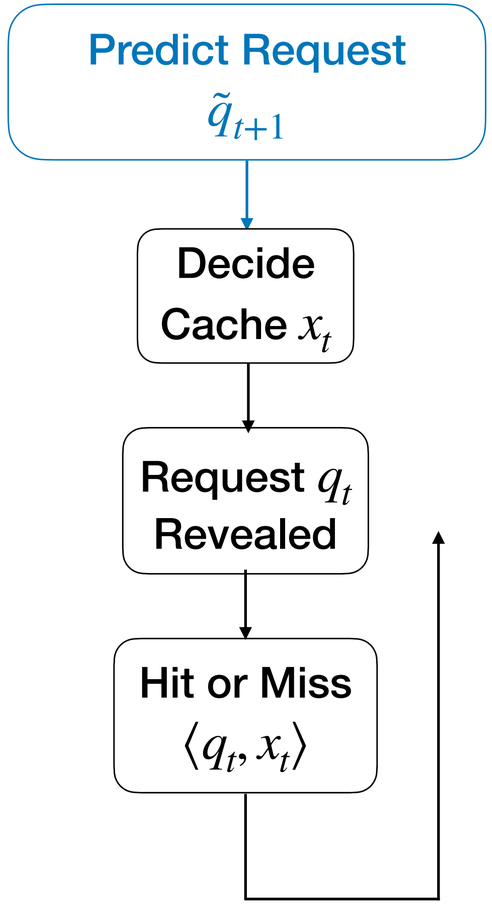
Result : $R_T = O\left(\sqrt{\sum_{t=1}^{T} \|c_t - \tilde{c}_t\|} \right)$

Worst case: $R_T = O\left(\sqrt{2T}\right)$ **Best case:** $R_T = 0$

Optimistic OCO - use case

- A library of Equal unit size N files, Cache with Capacity C.
- At each time step, a request vector is revealed $q_t \in \{0,1\}^N$.
- A prediction for the request is available. No guarantees on its quality.



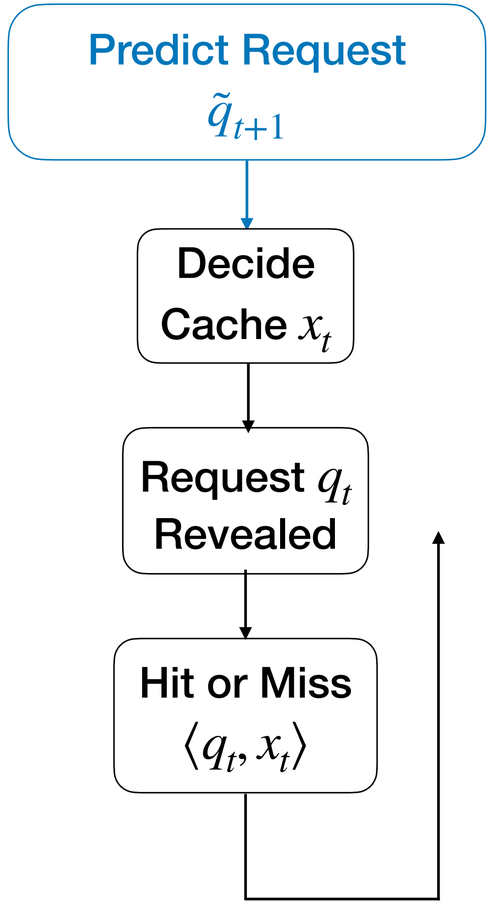


Optimistic OCO - use case

The guarantees of the solution Algorithm

- A library of Equal unit size N files, Cache with Capacity C.
- At each time step, a request vector is revealed $q_t \in \{0,1\}^N$.
- A prediction for the request is available. No guarantees on its quality.
- Update step: $x_{t+1} = \arg\min_{x \in \mathcal{X}} \left\{ \langle c_{1:t} + \tilde{c}_{t+1}, x \rangle + r_{1:t}(x) \right\}$
- Regularization: $r_t(x) \propto \sum_{i=1}^{t-1} \|c_i - \tilde{c}_i\| \times \|x\|$

Result : $R_T = O\left(\sqrt{\sum_{t=1}^{T} \|q_t - \tilde{q}_t\|} \right)$

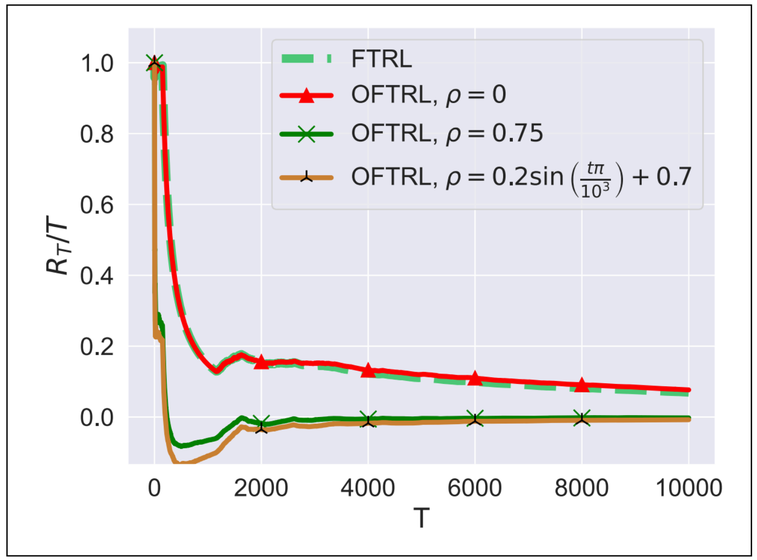


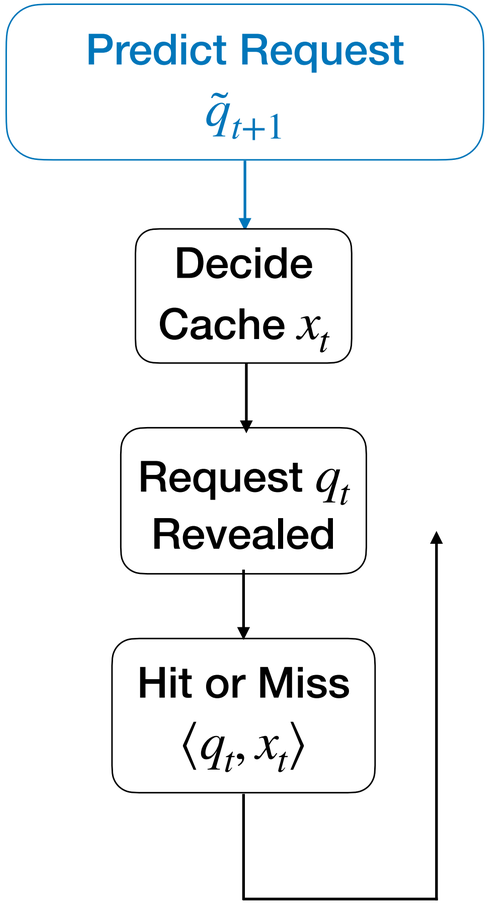
Optimistic OCO - use case

The guarantees of the solution Algorithm

- A library of Equal unit size N files, Cache with Capacity C.
- At each time step, a request vector is revealed $q_t \in \{0,1\}^N$.
- At prediction for the request is available. No guarantees on its quality.

Result : $R_T = O\left(\sqrt{\sum_{t=1}^{T} \|q_t - \tilde{q}_t\|}\right)$

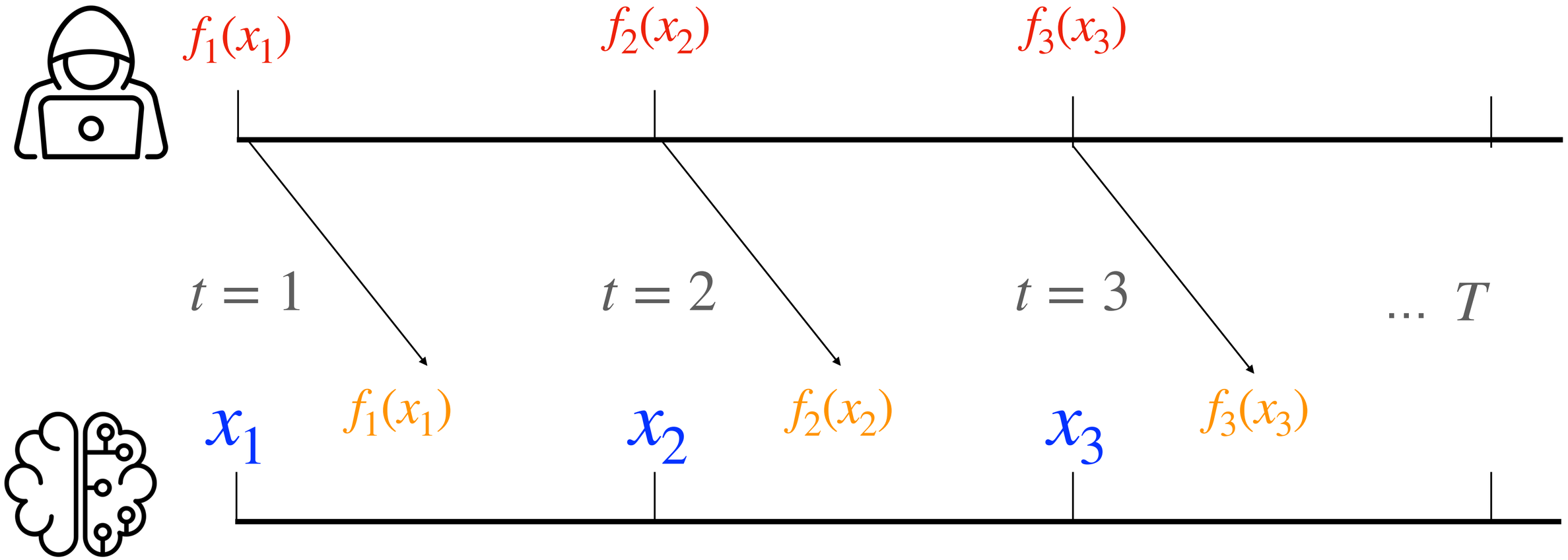




Back to the features we left aside: “Partial Feedback” & “States”

The Model of an Online *Adversarial Stateless* System

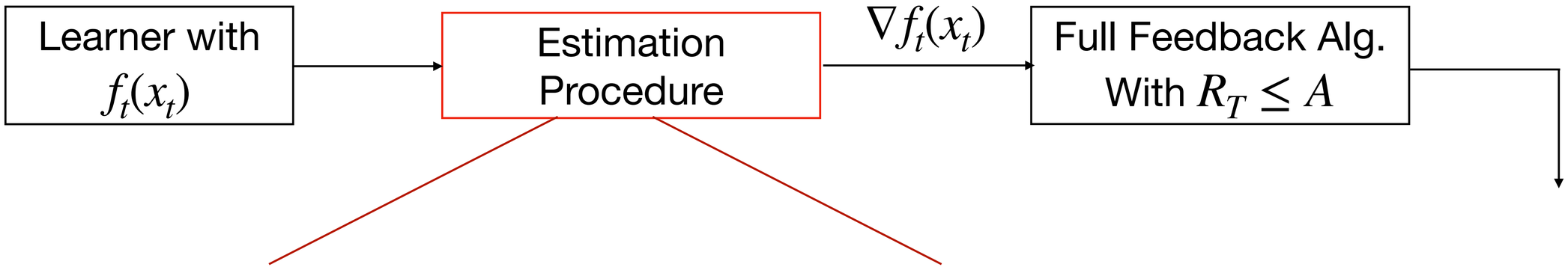
a.k.a adversarial bandits



The Model of an Online *Adversarial Stateless* System

Reduction to the full Feedback model

- Full feedback algorithms are used as a sub-routine.
- At a time step t, upon receiving: $f_t(x_t)$.



$\mathbb{E}[R_T] \leq$ no. actions $\times A$

The Model of an Online *Adversarial Stateless* System

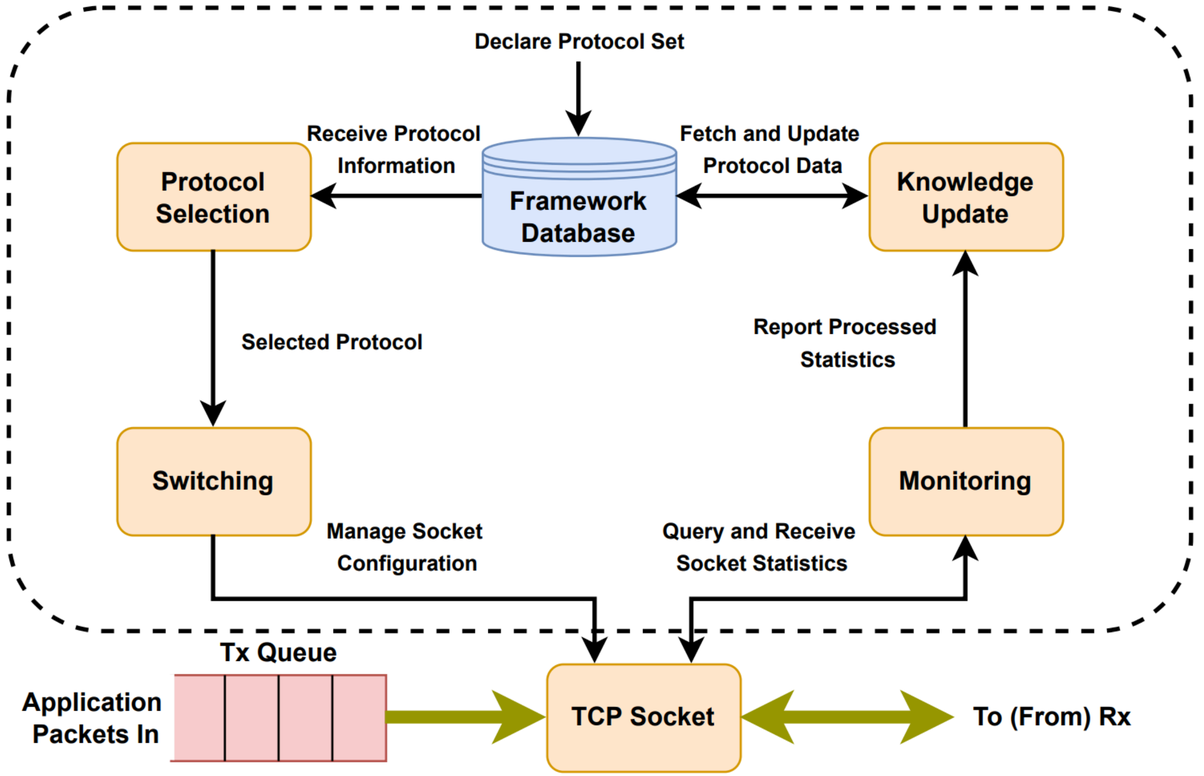
Find a random vector z_t such that. $\mathbb{E}[z_t] = \nabla f_t(x_t)$

- Consider a set of discrete actions. $[A] = \{a_0, a_1 \ldots, a_D\}$
- Make the decision variable $w_t \in \Delta_D$.
- Pick $p_t \sim w_t$
- The expected loss of the chosen arm is $f_t(w_t) = \langle c_t, w_t \rangle$
- After committing, we only see $c_t[p_t]$ (we don't have the full gradient)
- Construct the vector z_t s.t. $z_t^{[i]} = f_t(p_t)/w_t^{[i]}$ if $p_t = i$, or 0 otherwise.
- In the above construction $\mathbb{E}[z_t^{[i]}] = c_t^{[i]} = \nabla f_t(x_t)$

The Model of an Online *Adversarial Stateless* System

Use Case: Online TCP Protocol Selection

- The set of actions is a protocols set
- The reward function is “Delay” assumed unknown + adversarial.
- Partial feedback: only the delay of the selected protocol is revealed



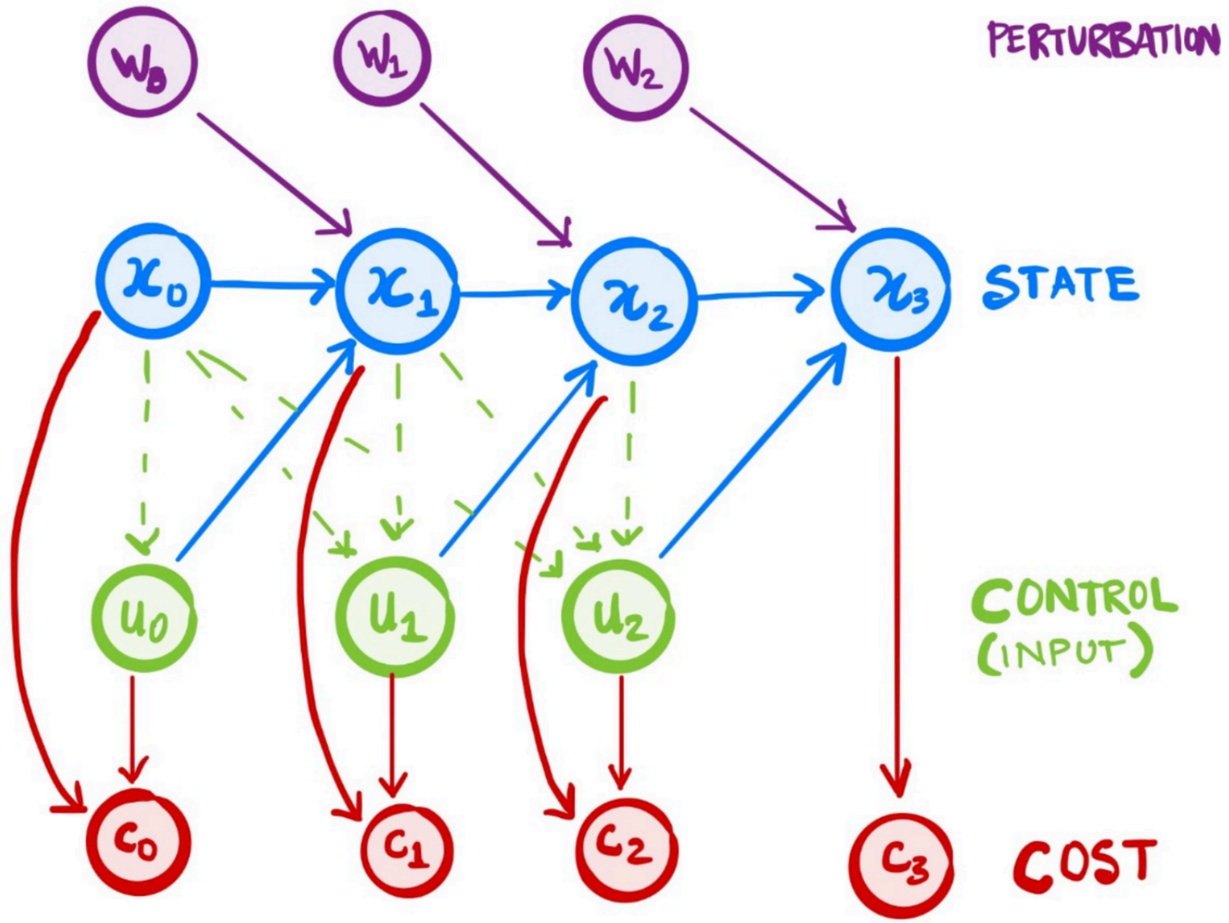
The Model of an Online *Adversarial* *Stateful* System

Model

State transition:

$$x_{t+1} = A_t x_t + B_t u_t + w_t$$

$$c_t(x_t, u_t)$$



Optimal Policy (only known in HS):

$$\min_{u(x)} \sum_{t=1}^{T} c_t(x_t, u_t)$$

$$\text{s.t.} \quad x_{t+1} = A_t x_t + B_t u_t + w_t$$

The Model of an Online *Adversarial* *Stateful* System

Metric

$$\max_{w_{1:T}} \left(\sum_{t=1}^{T} c_t(x_t, u_t) - \min_{\pi \in \Pi} \sum_{t=1}^{T} c_t(\hat{x}_t, \pi(\hat{x}_t)) \right)$$

- $\hat{x}_t$ = **counterfactual state sequence** under $\hat{u}_t = \pi(\hat{x}_t), \quad \hat{x}_{t+1} = A_t \hat{x}_t + B_t \hat{u}_t + w_t$
- Bounded noise $|w_t| \leq 1$

ICML 2020



https://sites.google.com/view/nsc-tutorial/home

The Model of an Online *Adversarial* *Stateful* System

Algorithm & Guarantee

Initialize $\vec{M} = M_1, \dots, M_H$

For $t = 1, \dots, T$ do

1. Use control $u_t = K_t x_t + \sum_{i \le H} M_i \, w_{t-i}$

2. Observe state x_{t+1}, compute noise $w_t = x_{t+1} - A_t x_t - B_t u_t$.

3. Construct cost function:

$$\ell_t(\vec{M}) = c_t\big(x_t(M_{1:H}), u_t(M_{1:H})\big)$$

4. Update $\vec{M}$

$$\vec{M} \leftarrow \vec{M} - \eta \, \nabla_{\vec{M}} \, \ell_t(\vec{M})$$

Efficient algorithm s.t.

$$\sum_{t=1}^{T} c_t(x_t, u_t) - \min_{\pi \in \Pi_{DAC}} \left(\sum_{t=1}^{T} c_t\big(\hat{x}_t, \pi(\hat{x}_t)\big) \right) \le O(\sqrt{T})$$

Conclusion

Highlights

- **Online Learning:** Time slotted, take actions to minimize future unknown costs.
- **RL:** Stateful, partial feedback system. Estimate from samples & act greedily.
- **Bandits:** RL with a single state.
- **OCO:** Stateless & Full-feedback. No estimation. Regularized optimization.
    - No-Regret FTRL solution algorithms, Optimism.
- **Adversarial Bandits:** OCO with partial feedback.
    - Reduction to full feedback.
- **Non-stochastic control:** Stateful Online learning. Adversarial transitions and costs.