FROM CONVEX OPTIMIZATION TO LEARNING IN GAMES

THEORY AND APPLICATIONS

Panayotis Mertikopoulos

French National Center for Scientific Research (CNRS)

Criteo AI Lab (CAIL)

⟨ ALMA graduate program | AGT + CVX/ML course | June 6, 2022 ⟩

Outline

1 Background & Motivation

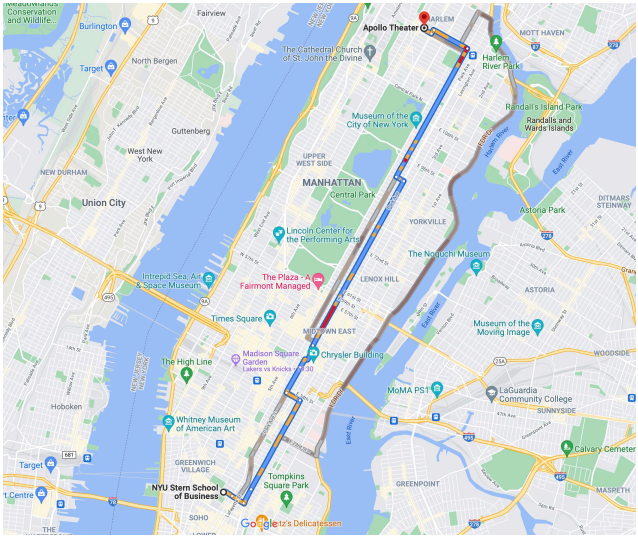
2 Theory: Mirror Descent

3 Applications: Traffic Routing

Game of roads

A beautiful morning commute in NYC

Game of roads



Manhattan at a glance

- 1, 632, 000 people
- 759, 000 daily trips
- Up to 10^4 requests/s
- 933 nodes
- 2950 edges
- 870, 000 O/D pairs
- $\approx 2 * 10^{16}$ paths

A very large game!

Online learning

A generic **online decision process:**

---

for each ***epoch*** and every ***player*** **do** # continuous / discrete # single- / multi

  Choose ***action*** # continuous / discrete

  Receive ***reward*** # endogenous / exogenous

  Get ***feedback*** (maybe) # full info / oracle / payoff-based

end for

Defining elements

- **Time:** continuous or discrete?
- **Players:** continuous or finite?
- **Actions:** continuous or finite?
- **Reward mechanism:** endogenous or exogenous (determined by other players or by "Nature")?
- **Feedback:** observe other actions / other rewards / only received?

Online learning

A generic **online decision process:**

---

for each ***epoch*** and every ***player*** **do** # continuous / discrete # single- / multi

  Choose ***action*** # continuous / discrete

  Receive ***reward*** # endogenous / exogenous

  Get ***feedback*** (maybe) # full info / oracle / payoff-based

end for

Defining elements

- **Time:** ~~continuous~~ or discrete
- **Players:** continuous ~~or finite~~
- **Actions:** ~~continuous~~ or finite
- **Reward mechanism:** endogenous or exogenous (determined by other players or by "Nature")?
- **Feedback:** observe other actions / other rewards / only received?

Outline

1 Background & Motivation

2 Theory: Mirror Descent

3 Applications: Traffic Routing

Problem setup

Primitives:

- ***Problem domain:*** convex subset $\mathcal{X}$ of $\mathcal{V}$
- ***Optimization objective:*** convex function $f: \mathcal{V} \to \mathbb{R} \cup \{\infty\}$ with $\mathrm{dom}\, f = \mathcal{X}$

Convex Optimization

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & x \in \mathcal{X} \end{aligned} \tag{Opt}$$

Problem setup

Primitives:

- ***Problem domain:*** convex subset $\mathcal{X}$ of $\mathcal{V}$
- ***Optimization objective:*** convex function $f: \mathcal{V} \to \mathbb{R} \cup \{\infty\}$ with $\mathbf{dom}\, f = \mathcal{X}$

Stochastic Convex Optimization

$$\begin{aligned} \text{minimize} \quad & f(x) = \mathbb{E}[F(x;\omega)] \\ \text{subject to} \quad & x \in \mathcal{X} \end{aligned} \tag{Stoch}$$

Problem setup

Primitives:

- ***Problem domain:*** convex subset $\mathcal{X}$ of $\mathcal{V}$
- ***Optimization objective:*** convex function $f: \mathcal{V} \to \mathbb{R} \cup \{\infty\}$ with $\mathbf{dom}\, f = \mathcal{X}$

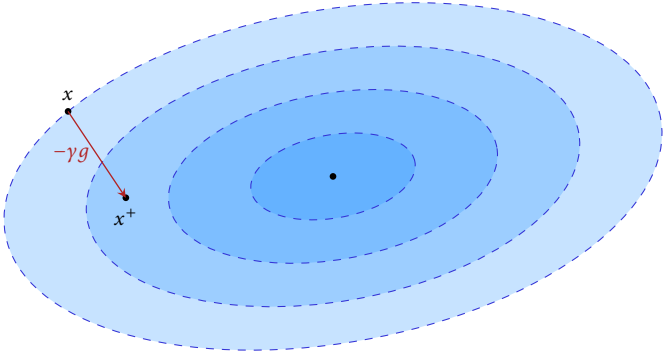
Online Convex Optimization

$$\begin{aligned} \text{minimize} \quad & f(x) = (1/T) \sum_{t=1}^{T} f_t(x) \\ \text{subject to} \quad & x \in \mathcal{X} \end{aligned} \tag{OCO}$$

Gradient methods in unconstrained problems

Gradient descent

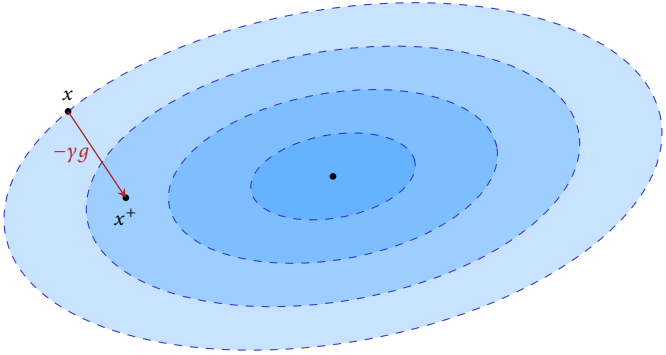
$$x_{t+1} = x_t - \gamma_t \nabla f(x_t) \tag{GD}$$



Gradient methods in unconstrained problems

Subgradient descent

$$x_{t+1} = x_t - \gamma_t g_t \qquad g_t \in \partial f(x_t) \qquad \text{(subGD)}$$



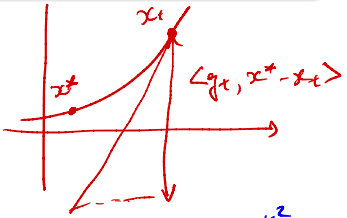
Projected gradient methods

Projected subgradient descent

$$x_{t+1} = \Pi_{\mathcal{X}}(x_t - \gamma_t g_t) \qquad \text{(PGD)}$$

$x_{t+1} = x_t - \gamma_t g_t \quad g_t = \nabla f(x_t)$

Let x^* be a solution of (Opt)



Let $D_t = \frac{1}{2}\|x_t - x^*\|^2$

$D_{t+1} = \frac{1}{2}\|x_t - \gamma g_t - x^*\|^2 = \underbrace{\frac{1}{2}\|x_t - x^*\|^2}_{D_t} - \gamma\langle g_t, x_t - x^*\rangle + \frac{\gamma^2}{2}\|g_t\|^2$

$f(\bar{x}_T) - f(x^*) \qquad \gamma\langle g_t, x_t - x^*\rangle \le D_t - D_{t+1} + \frac{\gamma^2}{2}\|g_t\|^2$

$\frac{1}{T}\sum_{t=1}^{T}\left[f(x_t) - f(x^*)\right] \le \frac{1}{T}\sum_{t=1}^{T}\langle g_t, x_t - x^*\rangle \le \frac{D_1}{\gamma T} + \frac{\gamma}{2T}\sum_{t=1}^{T}\|g_t\|^2$

Projected gradient methods

Projected subgradient descent

$$x_{t+1} = \Pi_{\mathcal{X}}(x_t - \gamma_t g_t) \qquad \text{(PGD)}$$

y_t

$y_{t+1} = x_t - \gamma_t g_t$

$x_{t+1} = \Pi_{\mathcal{X}}(y_{t+1})$

Lazy subgradient descent [Zinkevich, 2003]

$$y_{t+1} = y_t - \gamma_t g_t$$
$$x_{t+1} = \Pi_{\mathcal{X}}(y_{t+1}) \qquad \text{(LGD)}$$

Projected gradient methods

Projected subgradient descent

$$x_{t+1} = \Pi_{\mathcal{X}}(x_t - \gamma_t g_t) \tag{PGD}$$

Lazy subgradient descent [Zinkevich, 2003]

$$\begin{aligned} y_{t+1} &= y_t - \gamma_t g_t \\ x_{t+1} &= \Pi_{\mathcal{X}}(y_{t+1}) \end{aligned} \tag{LGD}$$

Dual averaging [Nesterov, 2009; Xiao, 2010]

$$\begin{aligned} y_{t+1} &= y_t - g_t \\ x_{t+1} &= \Pi_{\mathcal{X}}(\eta_{t+1} y_{t+1}) \end{aligned} \tag{DA}$$

Projected gradient methods

Lazy subgradient descent [Zinkevich, 2003]

$$y_{t+1} = y_t - \gamma_t g_t$$
$$x_{t+1} = \Pi_{\mathcal{X}}(y_{t+1}) \qquad \text{(LGD)}$$

Mirror descent

(Lazy) Mirror descent [à la Shalev-Shwartz, 2011; Nesterov, 2009]

$$y_{t+1} = y_t - \gamma_t g_t$$
$$x_{t+1} = Q(y_{t+1}) \qquad \text{(LMD)}$$

Mirror descent

(Lazy) Mirror descent [à la Shalev-Shwartz, 2011; Nesterov, 2009]

$$y_{t+1} = y_t - \gamma_t g_t$$
$$x_{t+1} = Q(y_{t+1}) \qquad \text{(LMD)}$$

Mirror map

Given a strictly convex ***regularizer*** $h: \mathcal{X} \to \mathbb{R}$, the ***mirror map*** $Q: \mathcal{V}^* \to \mathcal{X}$ is defined as

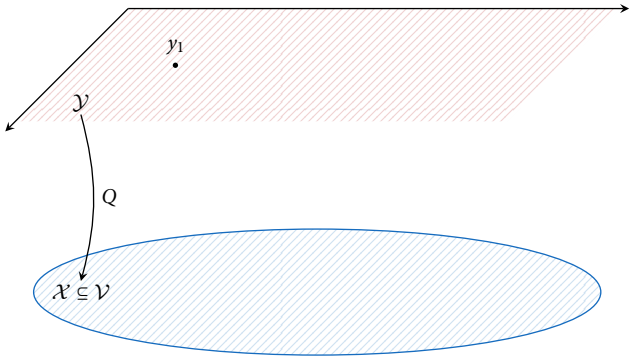
$$Q(y) = \arg\max_{x \in \mathcal{X}} \{\langle y, x \rangle - h(x)\}$$

LMD!

Lazy formulation of mirror descent [Shalev-Shwartz, 2011; Nesterov, 2009]

$$y_{t+1} = y_t - \gamma_t g_t$$
$$x_{t+1} = Q(y_{t+1}) \tag{DA}$$

where $Q(y) = \arg\max_{x \in \mathcal{X}} \{\langle y, x\rangle - h(x)\}$ is the ***mirror map*** associated to h

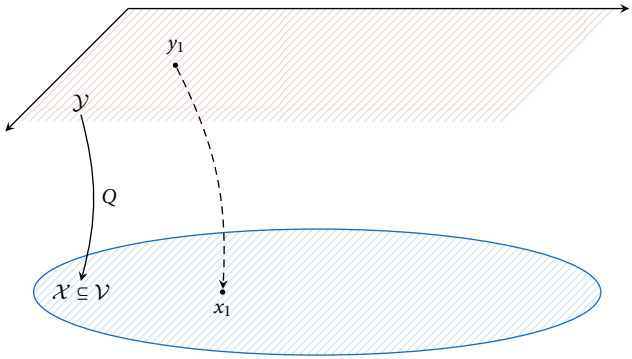


LMD!

Lazy formulation of mirror descent [Shalev-Shwartz, 2011; Nesterov, 2009]

$$y_{t+1} = y_t - \gamma_t g_t$$
$$x_{t+1} = Q(y_{t+1}) \qquad \text{(DA)}$$

where $Q(y) = \arg\max_{x \in \mathcal{X}} \{\langle y, x\rangle - h(x)\}$ is the ***mirror map*** associated to h

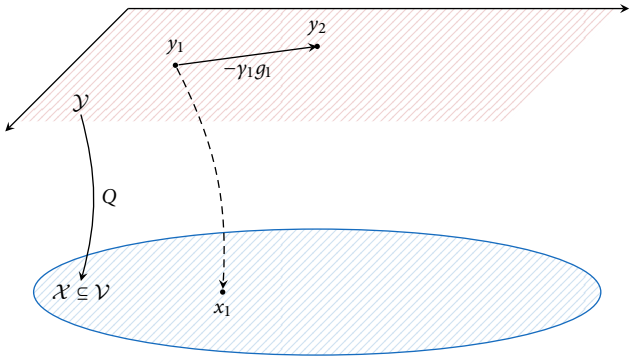


LMD!

Lazy formulation of mirror descent [Shalev-Shwartz, 2011; Nesterov, 2009]

$$y_{t+1} = y_t - \gamma_t g_t$$
$$x_{t+1} = Q(y_{t+1}) \qquad \text{(DA)}$$

where $Q(y) = \arg\max_{x \in \mathcal{X}} \{\langle y, x\rangle - h(x)\}$ is the ***mirror map*** associated to h

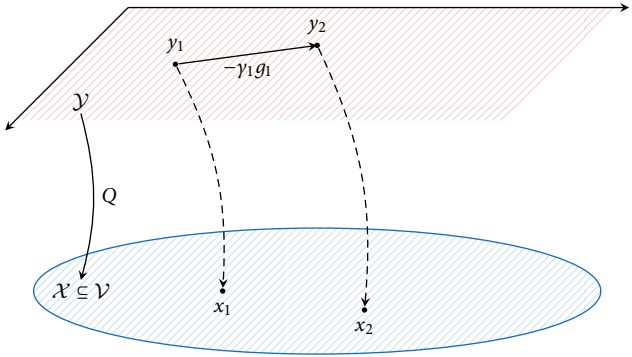


LMD!

Lazy formulation of mirror descent [Shalev-Shwartz, 2011; Nesterov, 2009]

$$y_{t+1} = y_t - \gamma_t g_t$$
$$x_{t+1} = Q(y_{t+1}) \qquad \text{(DA)}$$

where $Q(y) = \arg\max_{x \in \mathcal{X}} \{\langle y, x\rangle - h(x)\}$ is the ***mirror map*** associated to h

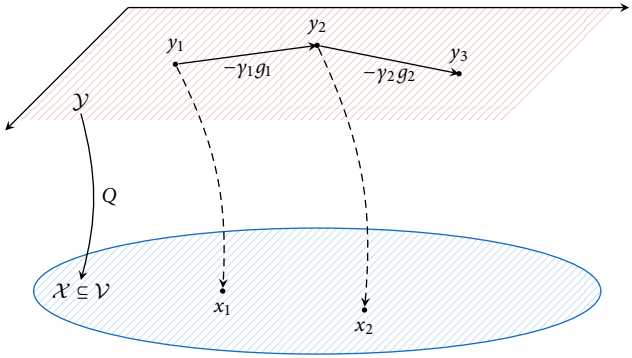


LMD!

Lazy formulation of mirror descent [Shalev-Shwartz, 2011; Nesterov, 2009]

$$y_{t+1} = y_t - \gamma_t g_t$$
$$x_{t+1} = Q(y_{t+1}) \qquad \text{(DA)}$$

where $Q(y) = \arg\max_{x \in \mathcal{X}} \{\langle y, x\rangle - h(x)\}$ is the ***mirror map*** associated to h

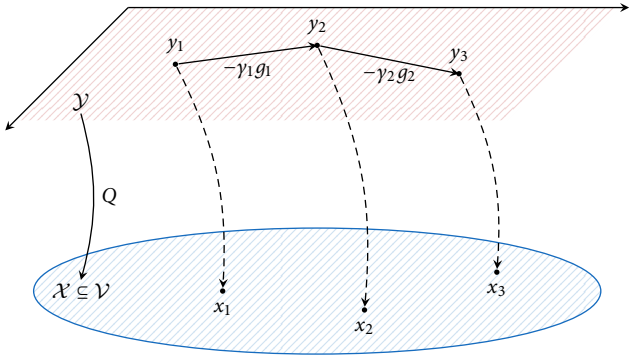


LMD!

Lazy formulation of mirror descent [Shalev-Shwartz, 2011; Nesterov, 2009]

$$y_{t+1} = y_t - \gamma_t g_t$$
$$x_{t+1} = Q(y_{t+1}) \qquad \text{(DA)}$$

where $Q(y) = \arg\max_{x \in \mathcal{X}} \{\langle y, x\rangle - h(x)\}$ is the ***mirror map*** associated to h



Mirror descent

Mirror descent [à la Nemirovski & Yudin, 1983; Beck & Teboulle, 2003]

$$x_{t+1} = P_{x_t}(-\gamma_t g_t) \tag{MD}$$

Mirror descent

Mirror descent [à la Nemirovski & Yudin, 1983; Beck & Teboulle, 2003]

$$x_{t+1} = P_{x_t}(-\gamma_t g_t) \tag{MD}$$

Prox-mapping

The ***prox-mapping*** of h is defined as

$$P_x(y) = \arg\min_{x' \in \mathcal{X}} \{\langle y, x - x' \rangle + D(x', x)\}$$

where the ***Bregman divergence*** D of h is given by

$$D(x', x) = h(x') - h(x) - \langle \nabla h(x), x' - x \rangle$$

Mirror descent

Mirror descent [à la Nemirovski & Yudin, 1983; Beck & Teboulle, 2003]

$$x_{t+1} = P_{x_t}(-\gamma_t g_t) \qquad \text{(MD)}$$

Prox-mapping

The ***prox-mapping*** of h is defined as

$$P_x(y) = \arg\min_{x' \in \mathcal{X}} \{\langle y, x - x' \rangle + D(x', x)\}$$

where the ***Bregman divergence*** D of h is given by

$$D(x', x) = h(x') - h(x) - \langle \nabla h(x), x' - x \rangle$$

Technical assumptions

- h is ***strongly convex*** $[h(x) - (K_h/2)\|x\|^2$ convex for some $K_h > 0]$
- ∂h admits a ***continuous selection*** [continuous $\nabla h(x) \in \partial h(x)$ for $x \in \operatorname{dom} \partial h$]

Example 1

Euclidean setup

- ***Problem domain:*** arbitrary
- ***Regularizer:*** $h(x) = \frac{1}{2}\|x\|_2^2$
- ***Bregman divergence:*** $D(x', x) = \frac{1}{2}\|x - x'\|_2^2$
- ***Mirror map:*** $Q(y) = \Pi_{\mathcal{X}}(y)$
- ***Prox-mapping:*** $P_x(y) = \Pi_{\mathcal{X}}(x + y)$

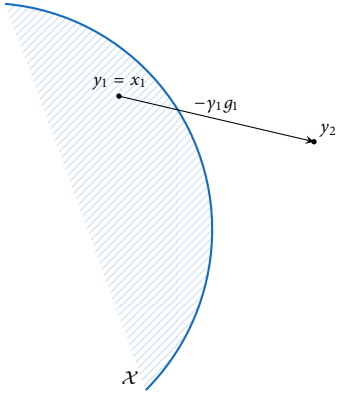
- ***Primal-dual variant:***

$$x_{t+1} = \Pi_{\mathcal{X}}(x_t - \gamma_t g_t) \tag{PGD}$$

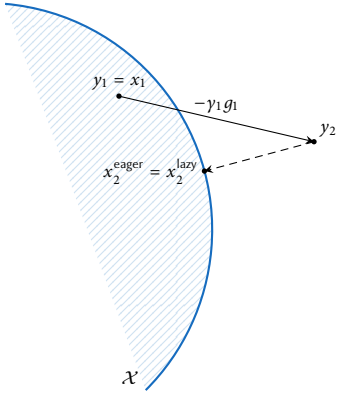
- ***Primal-dual variant:***

$$\begin{aligned} y_{t+1} &= y_t - \gamma_t g_t \\ x_{t+1} &= \Pi_{\mathcal{X}}(y_{t+1}) \end{aligned} \tag{LGD}$$

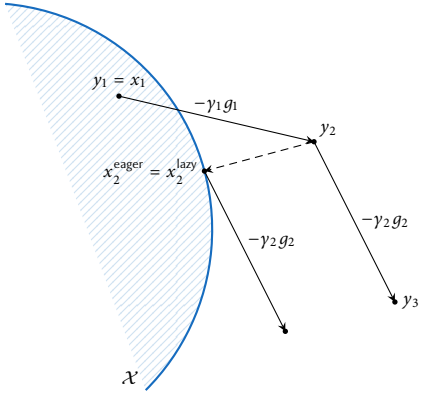
Lazy vs. eager



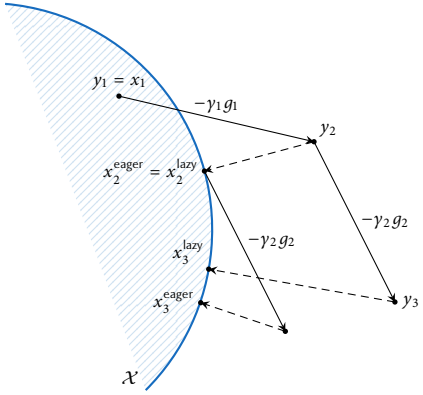
Lazy vs. eager



Lazy vs. eager



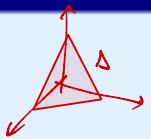
Lazy vs. eager



Example 2

Simplex setup

- ***Problem domain:*** simplex $\Delta_d = \{x \in \mathbb{R}^d_+ : \sum_{i=1}^d x_i = 1\}$ $[x_i \geq 0, \sum_i x_i = 1]$
- ***Regularizer:*** $h(x) = \sum_{i=1}^d x_i \log x_i$ [negative entropy]
- ***Bregman divergence:*** $D(x', x) = \sum_{i=1}^d x'_i \log(x'_i/x_i)$ [Kullback-Leibler divergence]
- ***Mirror map:*** $Q(y) = \Lambda(y) = \frac{(\exp(y_1),\dots,\exp(y_d))}{\sum_{i=1}^d \exp(y_i)}$ $Q(y) = \operatorname{argmax}_{x \in \Delta} \{\langle y, x\rangle - h(x)\}$ [logit map]
- ***Prox-mapping:*** $P_x(y) = \frac{(x_1 \exp(y_1),\dots,x_d \exp(y_d))}{\sum_{i=1}^d x_i \exp(y_i)}$



- ***Primal-dual variant:*** [Entropic gradient descent; Beck & Teboulle, 2003]

$$x_{t+1} = P_{x_t}(-\gamma_t g_t) \propto x_{i,t} \exp(-\gamma_t g_{i,t}) \quad \text{(EGD)}$$

- ***Primal-dual variant:*** [Exponential weights; Auer et al., 1995]

$$\begin{aligned} y_{t+1} &= y_t - \gamma_t g_t \\ x_{t+1} &= \Lambda(y_{t+1}) \propto \exp(y_{i,t+1}) \end{aligned} \quad \text{(EW)}$$

Example 3

Spectrahedron setup

- ***Problem domain:*** spectrahedron $[\mathbf{X} \succcurlyeq 0, \operatorname{tr}(\mathbf{X}) = 1]$
- ***Regularizer:*** $h(\mathbf{X}) = \operatorname{tr}[\mathbf{X} \log \mathbf{X}]$ [von Neumann entropy]
- ***Bregman divergence:*** $D(\mathbf{X}', \mathbf{X}) = \operatorname{tr}[\mathbf{X}'(\log \mathbf{X}' - \log \mathbf{X})]$ [quantum relative entropy]
- ***Mirror map:*** $Q(\mathbf{Y}) = \frac{\exp(\mathbf{Y})}{\operatorname{tr}[\exp(\mathbf{Y})]}$ [logit map]
- ***Prox-mapping:*** $P_{\mathbf{X}}(\mathbf{Y}) = \frac{\exp(\log \mathbf{X} + \mathbf{Y})}{\operatorname{tr}[\exp(\log \mathbf{X} + \mathbf{Y})]}$

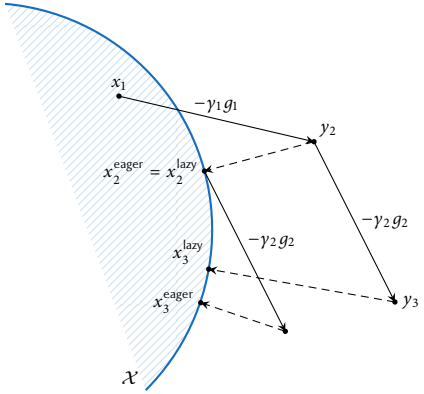
- ***Primal-dual variant:*** [Spectral gradient descent; Tsuda et al., 2005]

$$\mathbf{X}_{t+1} = \exp(\log \mathbf{X}_t - \gamma_t \mathbf{G}_t) \qquad \text{(specGD)}$$

- ***Primal-dual variant:*** [Matrix exponential learning; M. et al., 2017]

$$\mathbf{Y}_{t+1} = \mathbf{Y}_t - \gamma_t \mathbf{G}_t$$
$$\mathbf{X}_{t+1} = \frac{\exp(\mathbf{Y}_{t+1})}{\operatorname{tr}[\exp(\mathbf{Y}_{t+1})]} \qquad \text{(MXL)}$$

Lazy vs. eager



Equivalence of lazy and eager schemes

$$\operatorname{im} Q = \operatorname{ri} \mathcal{X} \implies \text{lazy} = \text{eager}$$

Blanket guarantees

Theorem (non-smooth; Nesterov, 2009; Shalev-Shwartz, 2011)

Assume:

- f is G-Lipschitz continuous
- (MD) is run for T steps with $\gamma = (1/G)\sqrt{R_h K_h / T}$ where $R_h = \max h - \min h$

Then: the "ergodic average" $\bar{x}_T = (1/T)\sum_{t=1}^{T} x_t$ enjoys the value convergence rate

$$f(\bar{x}_t) - \min f \leq 2G\sqrt{R_h/(K_h T)}$$

χ

Euclidean: $\chi = \Theta(\sqrt{d})$

Entropic: $\chi = \Theta(\sqrt{\log d})$

Theorem (smooth; Bauschke et al., 2017)

Assume:

- f is L-Lipschitz smooth relative to h [$Lh - f$ convex]
- (MD) is run for T steps with $\gamma \leq 1/L$

Then: x_t converges to a minimizer x^* of f at a rate of

$$f(x_t) - f(x^*) \leq \frac{LD(x^*, x_1)}{T}$$

ASSUMPTIONS:

(1) Lipschitz objective: $|f(x') - f(x)| \le G\|x' - x\|$ (LC)
Bounded gradients: $\|\nabla f(x)\| \le G$ (BG)

(2) Lipschitz smoothness: $f(x') \le f(x) + \langle \nabla f(x), x' - x \rangle + \frac{L}{2}\|x' - x\|^2$ (LS)
Lipschitz gradient: $\|\nabla f(x') - \nabla f(x)\| \le L\|x' - x\|$ (LG)

LOWER BOUNDS:

	OPT (Det / Static)	STOCH / ONLINE
LC / BG	$1/\sqrt{T}$	$1/\sqrt{T}$
LS / LG	$1/T^2$ Nemirovski 1979	$1/\sqrt{T}$
p-th level smoothness	$1/T^{\frac{3p+1}{2}}$ Nesterov 2015	$1/\sqrt{T}$ Tensor Methods, L-BFGS

Attained by Fast Grad Nesterov 1983

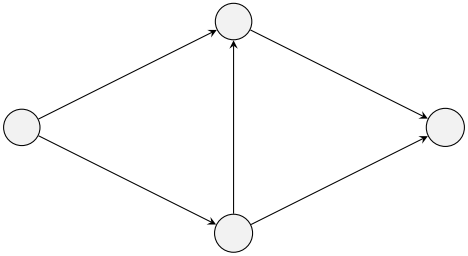
Outline

1 Background & Motivation

2 Theory: Mirror Descent

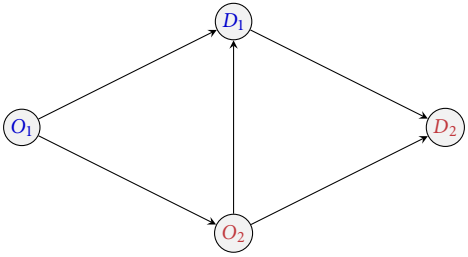
3 Applications: Traffic Routing

Nonatomic congestion games



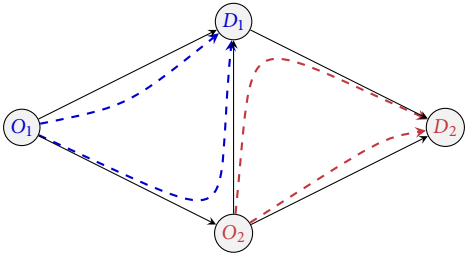
- ***Network:*** multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

Nonatomic congestion games



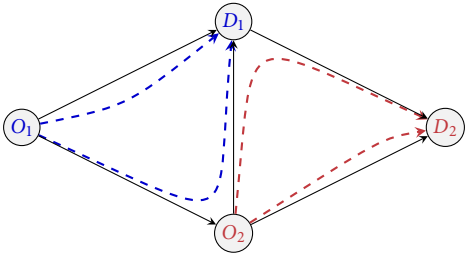
- ***Network:*** multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- ***O/D pairs*** $i \in \mathcal{N}$**:** origin O_i sends m_i units of traffic to destination D_i

Nonatomic congestion games



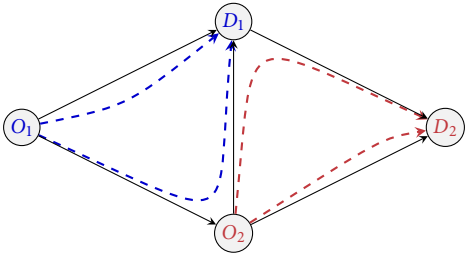
- ***Network:*** multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- ***O/D pairs*** $i \in \mathcal{N}$**:** origin O_i sends m_i units of traffic to destination D_i
- ***Paths*** $\mathcal{P}_i$**:** (sub)set of paths joining $O_i \rightsquigarrow D_i$

Nonatomic congestion games



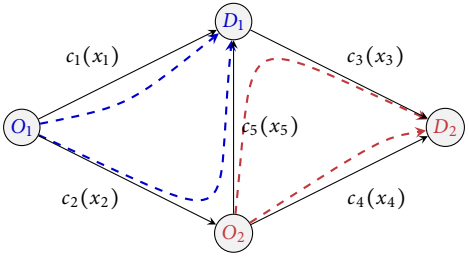
- ***Network:*** multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- ***O/D pairs*** $i \in \mathcal{N}$**:** origin O_i sends m_i units of traffic to destination D_i
- ***Paths*** $\mathcal{P}_i$**:** (sub)set of paths joining $O_i \rightsquigarrow D_i$
- ***Routing flow*** f_p**:** traffic along $p \in \mathcal{P} \equiv \bigcup_i \mathcal{P}_i$ generated by O/D pair owning p

Nonatomic congestion games



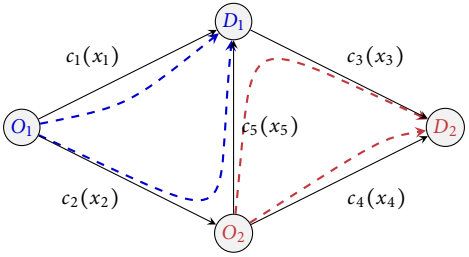
- ***Network:*** multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- ***O/D pairs*** $i \in \mathcal{N}$**:** origin O_i sends m_i units of traffic to destination D_i
- ***Paths*** $\mathcal{P}_i$**:** (sub)set of paths joining $O_i \rightsquigarrow D_i$
- ***Routing flow*** f_p**:** traffic along $p \in \mathcal{P} \equiv \bigcup_i \mathcal{P}_i$ generated by O/D pair owning p
- ***Load*** $x_e = \sum_{p \ni e} f_p$**:** total traffic along edge e

Nonatomic congestion games



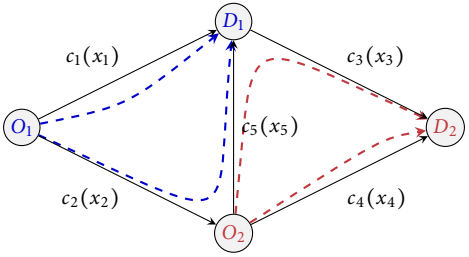
- ***Network:*** multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- ***O/D pairs*** $i \in \mathcal{N}$**:** origin O_i sends m_i units of traffic to destination D_i
- ***Paths*** $\mathcal{P}_i$**:** (sub)set of paths joining $O_i \rightsquigarrow D_i$
- ***Routing flow*** f_p**:** traffic along $p \in \mathcal{P} \equiv \bigcup_i \mathcal{P}_i$ generated by O/D pair owning p
- ***Load*** $x_e = \sum_{p \ni e} f_p$**:** total traffic along edge e
- ***Edge cost function*** $c_e(x_e)$**:** cost along edge e when edge load is x_e

Nonatomic congestion games



- ***Network:*** multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- ***O/D pairs*** $i \in \mathcal{N}$***:*** origin O_i sends m_i units of traffic to destination D_i
- ***Paths*** $\mathcal{P}_i$***:*** (sub)set of paths joining $O_i \rightsquigarrow D_i$
- ***Routing flow*** f_p***:*** traffic along $p \in \mathcal{P} \equiv \bigcup_i \mathcal{P}_i$ generated by O/D pair owning p
- ***Load*** $x_e = \sum_{p \ni e} f_p$***:*** total traffic along edge e
- ***Edge cost function*** $c_e(x_e)$***:*** cost along edge e when edge load is x_e
- ***Path cost:*** $c_p(f) = \sum_{e \in p} c_e(x_e)$

Nonatomic congestion games



- ***Network:*** multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- ***O/D pairs*** $i \in \mathcal{N}$***:*** origin O_i sends m_i units of traffic to destination D_i
- ***Paths*** $\mathcal{P}_i$***:*** (sub)set of paths joining $O_i \rightsquigarrow D_i$
- ***Routing flow*** f_p***:*** traffic along $p \in \mathcal{P} \equiv \bigcup_i \mathcal{P}_i$ generated by O/D pair owning p
- ***Load*** $x_e = \sum_{p \ni e} f_p$***:*** total traffic along edge e
- ***Edge cost function*** $c_e(x_e)$***:*** cost along edge e when edge load is x_e
- ***Path cost:*** $c_p(f) = \sum_{e \in p} c_e(x_e)$
- **Nonatomic congestion game:** $\mathcal{G} = (\mathcal{G}, \mathcal{N}, \{m_i\}_{i \in \mathcal{N}}, \{\mathcal{P}_i\}_{i \in \mathcal{N}}, \{c_e\}_{e \in \mathcal{E}})$

Traffic equilibrium

Wardrop equilibrium

The flow profile $f^* \in \mathcal{F}$ is a ***Wardrop equilibrium*** if

$$c_{p_i}(f^*) \leq c_{q_i}(f^*) \quad \text{for all utilized paths } p_i \in \mathcal{P}_i,\ i \in \mathcal{N} \tag{WE}$$

[**Equilibrium routing is envy-free:** all traffic elements experience the same latency]

Theorem (Beckmann et al., 1956)

$f^* \in \mathcal{F}$ is a Wardrop equilibrium if and only if it solves the convex problem

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in \mathcal{E}} \int_0^{x_e} c_e(w)\, dw \\ \text{subject to} \quad & x_e = \sum_{p \ni e} f_p,\ f \in \mathcal{F} \end{aligned} \tag{Eq}$$

The road to equilibrium

How to reach an equilibrium state?

- Standard rationality postulates $\rightsquigarrow$ meaningless [complete lack of knowledge]
- Recommender apps $\rightsquigarrow$ can lead to equilibrium [$\approx 10^8$ user base]

The road to equilibrium

How to reach an equilibrium state?

- Standard rationality postulates $\rightsquigarrow$ meaningless [complete lack of knowledge]
- Recommender apps $\rightsquigarrow$ can lead to equilibrium [$\approx 10^8$ user base]

Recommender must be able to solve in real time:

$$\begin{aligned} \text{minimize} \quad & L(f) = \sum_{e \in \mathcal{E}} \int_0^{x_e} c_e(w)\, dw \\ \text{subject to} \quad & x_e = \sum_{p \ni e} f_p, \ f \in \mathcal{F} \end{aligned} \tag{WE}$$

The road to equilibrium

How to reach an equilibrium state?

- Standard rationality postulates $\rightsquigarrow$ meaningless [complete lack of knowledge]
- Recommender apps $\rightsquigarrow$ can lead to equilibrium [$\approx 10^8$ user base]

Recommender must be able to solve in real time:

$$\begin{aligned} \text{minimize} \quad & L(f) = \sum_{e \in \mathcal{E}} \int_0^{x_e} c_e(w)\, dw \\ \text{subject to} \quad & x_e = \sum_{p \ni e} f_p, \ f \in \mathcal{F} \end{aligned} \tag{WE}$$

Challenges

- **Variability:** traffic conditions fluctuate unpredictably
- **Uncertainty:** congestion metrics only partially observable
- **Dimensionality:** exponential number of state variables

The model

Randomness and uncertainty:

- ***Exogenous randomness*** $\omega \in \Omega$ reflected in observed costs $\rightsquigarrow c_e(x_e; \omega)$

  ["State of the world": weather, accidents, added congestion...]

- ***Mean equilibrium flows***

$$\mathbb{E}_\omega[c_{p_i}(f^*; \omega)] \leq \mathbb{E}_\omega[c_{q_i}(f^*; \omega)] \quad \text{for all utilized paths } p_i \in \mathcal{P}_i,\ i \in \mathcal{N}$$

Sequence of events

1: **for all** $t = 1, 2, \ldots$ **do**
2: Interface recommends flow profile $x_t \in \mathcal{F}$
3: Nature determines state of the network $\omega_t \in \Omega$
4: Users on path p incur $c_p(x_t; \omega_t)$
5: **end for**

Equilibrium characterization

Stochastic convex programming characterization

f^* is a ***mean equilibrium flow*** if and only if it solves

$$\begin{aligned} \text{minimize} \quad & \tilde{L}(f) = \mathbb{E}\left[\sum_{e\in\mathcal{E}} \int_0^{x_e} c_e(w;\omega)\, dw\right] \\ \text{subject to} \quad & x_e = \sum_{p\ni e} f_p, \ f \in \mathcal{F} \end{aligned} \tag{Eq.S}$$

NB: ***Observed cost vectors*** $\rightsquigarrow$ ***stochastic gradients***

$$\nabla \tilde{L}(f) = \left(\tilde{c}_p(f)\right)_{p\in\mathcal{P}} = \mathbb{E}\Big[\left(c_p(f;\omega)\right)_{p\in\mathcal{P}}\Big]$$

Two sharply different frameworks:

- ***Static regime:*** ω_t remains constant with time
- ***Stochastic regime:*** ω_t fluctuates with time

Stochastic gradient descent

Stochastic gradient descent:

$$f_{t+1} = \mathrm{pr}_{\mathcal{F}}(f_t - \gamma \hat{c}_t) \qquad \text{(SGD)}$$

where $\hat{c}_t = c(f_t; \omega_t)$ is the ***cost profile*** at time t and $\gamma > 0$ is a ***step-size*** parameter

Theorem (folk)

If (SGD) *is run for* T *iterations with* $\gamma \propto 1/\sqrt{T}$*, the mean flow* $\bar{f}_T = T^{-1} \sum_{t=1}^{T} f_t$ *enjoys*

$$\mathbb{E}[\bar{L}(\bar{f}_T) - \min \bar{L}] = \mathcal{O}\left(\sqrt{P/T}\right)$$

Stochastic gradient descent

Stochastic gradient descent:

$$f_{t+1} = \mathrm{pr}_{\mathcal{F}}(f_t - \gamma \hat{c}_t) \qquad \text{(SGD)}$$

where $\hat{c}_t = c(f_t; \omega_t)$ is the ***cost profile*** at time t and $\gamma > 0$ is a ***step-size*** parameter

Theorem (folk)

If (SGD) *is run for* T *iterations with* $\gamma \propto 1/\sqrt{T}$*, the mean flow* $\bar{f}_T = T^{-1} \sum_{t=1}^{T} f_t$ *enjoys*

$$\mathbb{E}[\bar{L}(\bar{f}_T) - \min \bar{L}] = \mathcal{O}\left(\sqrt{P/T}\right)$$

Properties:

- ✓ **Optimal in T:** query complexity cannot be improved in the stochastic regime
- ✗ **Slow in P:** query complexity is **exponential** in the network's size
- ✗ **Non-adaptive:** requires tuning of γ
- ✗ **Offline:** $\bar{f}_t$ is never recommended

Exponential weights

An idea from the multi-armed bandits literature [Auer et al., 1995]

- Keep a score for each path, based on its performance so far
- Allocate traffic proportionally to the exponential of this score

Algorithm Exponential weights (ExpWeight)

Require: horizon T; step-size $\gamma > 0$

Initialize score vector $y \in \mathbb{R}^{\mathcal{P}}$

1: **for all** $t = 1, 2, \dots T$ **do**
2: Route according to $f_t \sim \exp(y_t)$ # routing recommendation
3: Observe cost profile: $\hat{c}_t \leftarrow (c_p(f_t; \omega_t))_{p \in \mathcal{P}}$ # cost feedback
4: Update path scores: $y_{t+1} \leftarrow y_t - \gamma \hat{c}_t$ # update step
5: **end for**
6: **return** $\bar{f}_T = (1/T) \sum_{t=1}^{T} f_t$ # output flow

ExpWeight guarantees

Theorem (folk-ish)

If ExpWeight *is run for T steps with $\gamma \propto 1/\sqrt{T}$, the mean flow $\bar{f}_T = T^{-1} \sum_{t=1}^{T} f_t$ enjoys*

$$\bar{L}(\bar{f}_T) - \min \bar{L} = \mathcal{O}\left(\sqrt{\log P / T}\right)$$

ExpWeight guarantees

Theorem (folk-ish)

If ExpWeight *is run for T steps with $\gamma \propto 1/\sqrt{T}$, the mean flow $\bar{f}_T = T^{-1}\sum_{t=1}^{T} f_t$ enjoys*

$$\bar{L}(\bar{f}_T) - \min \bar{L} = \mathcal{O}\left(\sqrt{\log P / T}\right)$$

Properties:

- ✓ **Optimal in T:** query complexity cannot be improved in the stochastic regime
- ✓ **Optimal in P:** query complexity is **polynomial** in the network's size
- ✗ **Non-adaptive:** requires tuning of γ
- ✗ **Offline:** $\bar{f}_t$ is never recommended

The static case

Is the situation the same in static the static regime?

- ✓ Nesterov's accelerated gradient (NAG) method achieves $\mathcal{O}(1/T^2)$ in static programs
- ✗ But **exponential dependence** on $|\mathcal{G}|$

Can we get rates that are optimal in both T and P?

The static case

Is the situation the same in static the static regime?

- ✓ Nesterov's accelerated gradient (NAG) method achieves $\mathcal{O}(1/T^2)$ in static programs
- ✗ But **exponential dependence** on $|\mathcal{G}|$

Can we get rates that are optimal in both T and P?

Algorithm Accelerated exponential weights (AcceleWeight) [Vu et al., 2021]

Require: initial score vector $y_0 \leftarrow 0$; moving weight $\alpha_0 \leftarrow 0$; step $\gamma_0 \leftarrow 1/(NM\beta)$ [$\beta \rightsquigarrow$ Lipschitz modulus]

1: **for all** $t = 1, 2, \dots T$ **do**
2: set $z_t \propto \exp(y_{t-1})$ # ExpWeight step
3: set $x_t \leftarrow \alpha_{t-1}x_{t-1} + (1-\alpha_{t-1})z_t$ # Nesterov momentum
4: set $\gamma_t \leftarrow \frac{1}{2}[2\gamma_{t-1} + \gamma_0 + \sqrt{4\gamma_{t-1}\gamma_0 + \gamma_0^2}]$ # NAG step-size
5: set $\alpha_t \leftarrow \gamma_{t-1}/\gamma_t$ # moving weight update
6: set $\tilde{z}_t \leftarrow \alpha_t x_t + (1-\alpha_t)z_t$ and get $c_t \leftarrow c(\tilde{z}_t)$ # route and measure costs
7: set $y_t \leftarrow y_{t-1} - (1-\alpha_t)\gamma_t c_t$ # update path scores
8: **end for**
9: **return** x_t # output flow

AcceleWeight guarantees

Theorem (Vu, Antonakopoulos & M, NeurIPS 2021)

In the static regime, AcceleWeight *enjoys the rate of convergence*

$$L(f_T) - \min L \leq \frac{4\beta^2 N^2 M^2 \log P}{T^2} = \mathcal{O}\left(\frac{\log P}{T^2}\right)$$

AcceleWeight guarantees

Theorem (Vu, Antonakopoulos & M, NeurIPS 2021)

In the static regime, AcceleWeight *enjoys the rate of convergence*

$$L(f_T) - \min L \leq \frac{4\beta^2 N^2 M^2 \log P}{T^2} = \mathcal{O}\left(\frac{\log P}{T^2}\right)$$

Properties:

- ✓ **Optimal in T:** query complexity cannot be improved in the **static** regime
- ✓ **Optimal in P:** query complexity is **polynomial** in the network's size
- ✗ **Non-adaptive:** requires tuning of γ
- ✗ **Offline:** f_t is never recommended

The good

The good:

- ✓ In the stochastic regime, ExpWeight is **optimal** in T and P
- ✓ In the static regime, AcceleWeight is **optimal** in T and P

The good, *the bad*

The good:

- ✓ In the stochastic regime, ExpWeight is **optimal** in T and P
- ✓ In the static regime, AcceleWeight is **optimal** in T and P

The bad:

- In the static regime, ExpWeight is **very slow** in T
- In the stochastic regime, AcceleWeight **does not converge**

The good, *the bad*, and the ugly

The good:

- ✓ In the stochastic regime, ExpWeight is **optimal** in T and P
- ✓ In the static regime, AcceleWeight is **optimal** in T and P

The bad:

- In the static regime, ExpWeight is **very slow** in T
- In the stochastic regime, AcceleWeight **does not converge**

The ugly:

- ✗ Tuning the step-size is impractical / impossible
- ✗ Output is never recommended

Adaptive algorithms

Compare observations:

- In the static regime: $\|c_{t+1} - c_t\|_\infty$ should become small over time
- In the stochastic regime: $\|c_{t+1} - c_t\|_\infty$ remains bounded away from zero

Adaptive algorithms

Compare observations:

- In the static regime: $\|c_{t+1} - c_t\|_\infty$ should become small over time
- In the stochastic regime: $\|c_{t+1} - c_t\|_\infty$ remains bounded away from zero

Adaptive step-size (Antonakopoulos & M, 2021; Hsieh, Antonakopoulos & M, COLT 2021)

$$\gamma_t = \frac{1}{\sqrt{1 + \sum_{s=1}^{t-1} \|c_{s+1} - c_s\|_\infty^2}} \qquad \text{(Adapt)}$$

Adaptive algorithms

Compare observations:

- In the static regime: $\|c_{t+1} - c_t\|_\infty$ should become small over time
- In the stochastic regime: $\|c_{t+1} - c_t\|_\infty$ remains bounded away from zero

Adaptive step-size (Antonakopoulos & M, 2021; Hsieh, Antonakopoulos & M, COLT 2021)

$$\gamma_t = \frac{1}{\sqrt{1 + \sum_{s=1}^{t-1} \|c_{s+1} - c_s\|_\infty^2}} \qquad \text{(Adapt)}$$

Algorithm ExpWeight + Adapt [Antonakopoulos & M, 2021]

Initialize score vector $y \in \mathbb{R}^{\mathcal{P}}$

1: **for all** $t = 1, 2, \dots T$ **do**
2: Route according to $f_t \sim \exp(y_t)$ # ExpWeight update
3: Observe cost profile: $\hat{c}_t \leftarrow (c_p(f_t; \omega_t))_{p \in \mathcal{P}}$ # cost feedback
4: Update path scores: $y_{t+1} \leftarrow y_t - \gamma_t \hat{c}_t$ # Adapt step
5: **end for**
6: **return** $\bar{f}_T = (1/T) \sum_{t=1}^T f_t$ # output flow

Guarantees of ExpWeight + Adapt

Theorem (Antonakopoulos & M, NeurIPS 2021)

Suppose that ExpWeight +Adapt *is run for T steps. Then $\bar{f}_T$ enjoys the rate*

$$\mathbb{E}[\bar{L}(\bar{f}_T) - \min \bar{L}] = \mathcal{O}\left(\frac{\log(PT)}{T} + \sigma\sqrt{\frac{\log(PT)}{T}}\right)$$

where σ^2 is the variance of $\|c'(x;\omega)\|_{\mathcal{L}^1}$.

Guarantees of ExpWeight + Adapt

Theorem (Antonakopoulos & M, NeurIPS 2021)

Suppose that ExpWeight +Adapt *is run for T steps. Then $\bar{f}_T$ enjoys the rate*

$$\mathbb{E}[\bar{L}(\bar{f}_T) - \min \bar{L}] = \mathcal{O}\left(\frac{\log(PT)}{T} + \sigma\sqrt{\frac{\log(PT)}{T}}\right)$$

where σ^2 is the variance of $\|c'(x;\omega)\|_{\mathcal{L}^1}$.

Properties:

- ✓ **Optimal in stochastic regime:** query complexity cannot be improved in T if $\sigma > 0$
- ▸ **Better** than ExpWeight in the static regime, but **worse** than AcceleWeight
- ✓ **Adaptive:** no hyperparameter tuning required
- ✗ **Offline:** $\bar{f}_t$ is never recommended

AdaWeight

Is there a path to universal acceleration?

AdaWeight

Is there a path to universal acceleration?

Algorithm Adaptive exponential weights (AdaWeight) [Vu et al., 2021]

Initialize score vector $y_1 \leftarrow 0$; moving weight $\alpha_0 \leftarrow 0$; step $\eta_1 \leftarrow 1$

for all $t = 1, 2, \dots T$ **do**

- set $z_t \propto \exp(\eta_t y_t)$ # ExpWeight step
- set $\tilde{z} \leftarrow \left(\alpha_t z_t + \sum_{s=0}^{t-1} \alpha_s z_{s+1/2}\right) \Big/ \sum_{s=0}^{t} \alpha_s$ and get $\tilde{c}_t \leftarrow c(\tilde{z}_t; \omega_t)$ # reweigh + explore
- set $y_{t+1/2} \leftarrow y_t - \alpha_t \tilde{c}_t$ # score update
- set $z_{t+1/2} \propto \exp(\eta_t y_{t+1/2})$ # ExpWeight step
- set $x_t \leftarrow \left(\sum_{s=0}^{t} \alpha_s z_{s+1/2}\right) \Big/ \sum_{s=0}^{t} \alpha_s$ and get $c_t \leftarrow c(x_t; \omega_t)$ # route and measure costs
- set $y_{t+1} \leftarrow y_t - \gamma_t c_t$ # update scores
- set $\eta_{t+1} \leftarrow \eta_t \sqrt{1 + \alpha_t^2 \|c_t - \tilde{c}_t\|_\infty^2}$ # Adapt step

end for

return x_t # output flow

[Borrows ideas from ExpWeight + NAG + extra-gradient + dual extrapolation methods]

AdaWeight guarantees

Theorem (Vu et al., 2021)

AdaWeight *enjoys the rate of convergence*

$$\mathbb{E}[L(f_T) - \min L] = \mathcal{O}\left(\frac{\log P}{T^2} + \frac{\sigma \log P}{\sqrt{T}}\right)$$

AdaWeight guarantees

Theorem (Vu et al., 2021)

AdaWeight *enjoys the rate of convergence*

$$\mathbb{E}[L(f_T) - \min L] = \mathcal{O}\left(\frac{\log P}{T^2} + \frac{\sigma \log P}{\sqrt{T}}\right)$$

Properties:

- ✓ **Optimal in stochastic regime:** query complexity cannot be improved in T if $\sigma > 0$
- ✓ **Optimal in static regime:** query complexity cannot be improved in T if $\sigma = 0$
- ✓ **Fast in P:** query complexity is **polynomial** in the network's size
- ✓ **Adaptive:** does not require any tuning or prior system knowledge
- ✓ **Online:** guarantees concern the recommended flows

AdaWeight in practice

Numerical experiments in the Anaheim metropolitan area

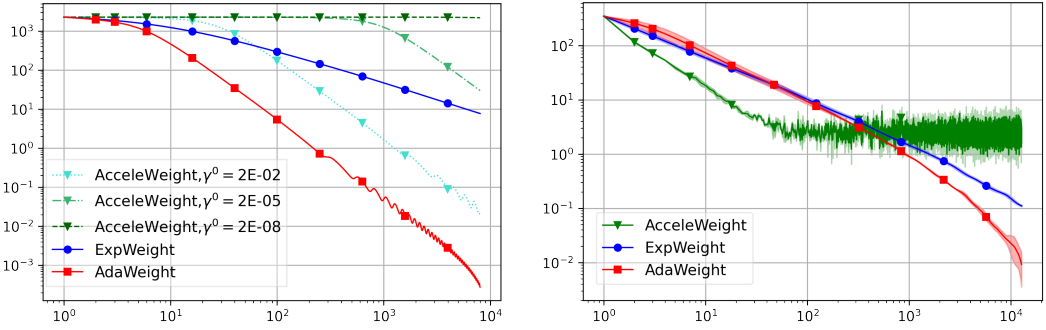


Figure: ExpWeight, AcceleWeight & AdaWeight in static (left) and stochastic (right) conditions

UnderGrad: The theory under the hood

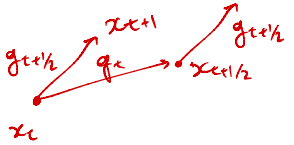
Is there a path to universal acceleration ***for arbitrary domains?***

UnderGrad: The theory under the hood

Is there a path to universal acceleration *for arbitrary domains?*

Dual extrapolation (DE)

$$\begin{aligned} y_{t+1/2} &= y_t - \gamma_t g_t & x_{t+1/2} &= Q(\eta_t y_{t+1/2}) \\ y_{t+1} &= y_t - \gamma_t g_{t+1/2} & x_{t+1} &= Q(\eta_{t+1} y_{t+1}) \end{aligned} \qquad \text{(DE)}$$



UnderGrad: The theory under the hood

Is there a path to universal acceleration ***for arbitrary domains?***

Dual extrapolation (DE)

$$\begin{aligned} y_{t+1/2} &= y_t - \gamma_t g_t & x_{t+1/2} &= Q(\eta_t y_{t+1/2}) \\ y_{t+1} &= y_t - \gamma_t g_{t+1/2} & x_{t+1} &= Q(\eta_{t+1} y_{t+1}) \end{aligned} \tag{DE}$$

Adaptive learning rate

$$\eta_{t+1} = \sqrt{\frac{K_h(R_h + K_h\|\mathcal{X}\|^2)}{K_h + \sum_{s=1}^{t} \gamma_s^2 \|g_{s+1/2} - g_s\|^2}} \tag{Adapt}$$

UnderGrad: The theory under the hood

Is there a path to universal acceleration *for arbitrary domains?*

Dual extrapolation (DE)

$$\begin{aligned} y_{t+1/2} &= y_t - \gamma_t g_t & x_{t+1/2} &= Q(\eta_t y_{t+1/2}) \\ y_{t+1} &= y_t - \gamma_t g_{t+1/2} & x_{t+1} &= Q(\eta_{t+1} y_{t+1}) \end{aligned} \tag{DE}$$

Adaptive learning rate

$$\eta_{t+1} = \sqrt{\frac{K_h(R_h + K_h\|\mathcal{X}\|^2)}{K_h + \sum_{s=1}^t \gamma_s^2 \|g_{s+1/2} - g_s\|^2}} \tag{Adapt}$$

Iterate averaging

$$\tilde{x}_t = \frac{\gamma_t x_t + \sum_{s=1}^{t-1} \gamma_s x_{s+1/2}}{\sum_{s=1}^t \gamma_s}$$

$$\tilde{x}_{t+1/2} = \frac{\gamma_t x_{t+1/2} + \sum_{s=1}^{t-1} \gamma_s x_{s+1/2}}{\sum_{s=1}^t \gamma_s}$$

UnderGrad: The theory under the hood

Is there a path to universal acceleration *for arbitrary domains?*

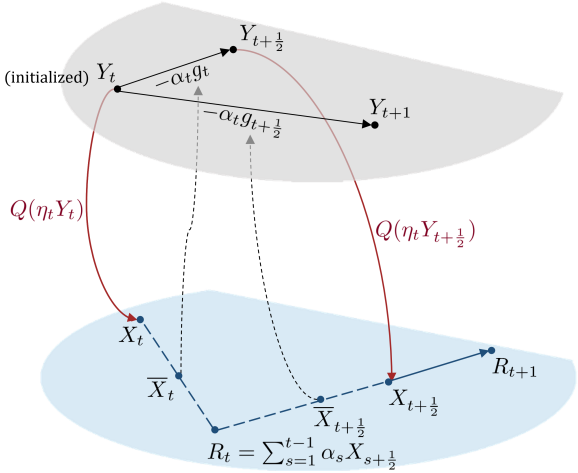


Figure: The UnderGrad algorithm

UnderGrad: The theory under the hood

Is there a path to universal acceleration *for arbitrary domains?*

Theorem (Antonakopoulos, Vu, Cevher, Levy & M, ICML 2022)

Suppose that UnderGrad *is run for T iterations with $\gamma_t = t$. Then the algorithm's output state $\bar{x}_T \equiv \bar{x}_{T+1/2}$ concurrently enjoys the following guarantees:*

a) If f satisfies (LC)/(BG), then

$$\mathbb{E}[f(\bar{x}_T) - \min f] \leq 2C_h \sqrt{\frac{K_h + 8(G^2 + \sigma^2)}{K_h T}}$$

b) If f satisfies (LS)/(LG), then

$$\mathbb{E}[f(\bar{x}_T) - \min f] \leq \frac{32\sqrt{2}C_h^2 L}{K_h T^2} + \frac{8\sqrt{2}C_h \sigma}{\sqrt{K_h T}}$$

where $C_h = \sqrt{R_h + K_h \|\mathcal{X}\|^2}$.

References I

Antonakopoulos, K. and Mertikopoulos, P. Adaptive first-order methods revisited: Convex optimization without Lipschitz requirements. In *NeurIPS '21: Proceedings of the 35th International Conference on Neural Information Processing Systems*, 2021.

Antonakopoulos, K., Vu, D. Q., Cevher, V., Levy, K. Y., and Mertikopoulos, P. Scaling up universal methods for convex optimization. In *ICML '22: Proceedings of the 39th International Conference on Machine Learning*, 2022.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, 1995.

Bauschke, H. H., Bolte, J., and Teboulle, M. A descent lemma beyond Lipschitz gradient continuity: First-order methods revisited and applications. *Mathematics of Operations Research*, 42(2):330–348, May 2017.

Beck, A. and Teboulle, M. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31 (3):167–175, 2003.

Beckmann, M., McGuire, C. B., and Winsten, C. *Studies in the Economics of Transportation*. Yale University Press, 1956.

Hsieh, Y.-G., Antonakopoulos, K., and Mertikopoulos, P. Adaptive learning in continuous games: Optimal regret bounds and convergence to Nash equilibrium. In *COLT '21: Proceedings of the 34th Annual Conference on Learning Theory*, 2021.

Mertikopoulos, P., Belmega, E. V., Negrel, R., and Sanguinetti, L. Distributed stochastic optimization via matrix exponential learning. *IEEE Trans. Signal Process.*, 65(9):2277–2290, May 2017.

Nemirovski, A. S. and Yudin, D. B. *Problem Complexity and Method Efficiency in Optimization*. Wiley, New York, NY, 1983.

Nesterov, Y. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, 2009.

Shalev-Shwartz, S. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2011.

References II

Tsuda, K., Rätsch, G., and Warmuth, M. K. Matrix exponentiated gradient updates for on-line Bregman projection. *Journal of Machine Learning Research*, 6:995-1018, 2005.

Vu, D. Q., Antonakopoulos, K., and Mertikopoulos, P. Fast routing under uncertainty: Adaptive learning in congestion games with exponential weights. In *NeurIPS '21: Proceedings of the 35th International Conference on Neural Information Processing Systems*, 2021.

Xiao, L. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11: 2543-2596, October 2010.

Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In *ICML '03: Proceedings of the 20th International Conference on Machine Learning*, pp. 928-936, 2003.

Games in Grenoble

 # If you like mountains and/or games, drop me an e-mail (doc / post-doc level)

In Game Theory:

$$u_k(x_k; x_{-k}) = \sum_{a_k \in A_k} \sum_{a_{-k} \in A_{-k}} x_{k,a_k} \, x_{-k,a_{-k}} \, u_k(a_k; a_{-k})$$

Exponential Gradient Descent

$$y_{a,t+1} = y_{a,t} + \gamma \, u(a; x_{-k,t})$$

$$\Downarrow$$

$$y_{a,t} = \sum_{s=1}^{t} \text{payoff}(a,s)$$