Improving Adaptive Online Learning Using PDEs

Zhiyu Zhang
BU SIAM Chapter Seminar

October 3, 2023

Overall of the talk

Part 1. What is (adversarial) online learning?

– A two-player repeated game.

Part 2. What is adaptivity in online learning?

– Key idea: minimax optimality $\Longrightarrow$ instance optimality

– Bayesian interpretation

Part 3. How to achieve (comparator) adaptivity?

– We can use PDEs to make the workflow easier!

Part 1 – What is (adversarial) online learning?

Sequential decision making under uncertainty

We, as a decision making agent, repeatedly

- make a decision;
- receive a feedback from an uncertain environment;
- suffer a loss.

Uncertainty: the effect of a decision is not fully known before we apply it.

Exactly why the problem is practical and interesting!

A few examples

Example	Weather forecasting	Investment	Robotic control
Decision	tomorrow’s weather	amount we buy	direction of the car
Feedback	actual weather	price change	road condition
Loss / reward	forecasting accuracy	wealth increase	safety margin

Why the name “Online Learning”?

Online:

The data generation mechanism is sequential and possibly interactive.

Learning:

Our “overall performance” (?) should gradually improve as we observe more data.

In plain English, “learning” a skill means we get better doing it!

How do we model the uncertainty of the environment?

– Known distribution? Often just computation, nothing very special...

– Statistical model (e.g., IID, Markov process, linear system,...) with unknown components?

Statistical machine learning – statisticians love that...

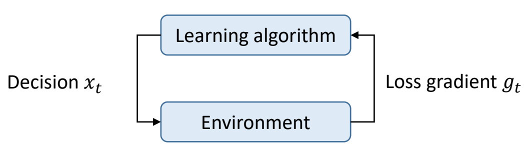
– A statistics-free model? After all, nature can be adversarial.

Game theory – this is our focus.

Motivated by the taste of many CS people: they favor the "weakest assumptions" possible.

Online learning as a game

Online Convex Optimization [Zinkevich, '03] is a two-player repeated game. In each round,



1. we pick a decision x_t in a closed convex set $\mathcal{X} \subset \mathbb{R}^d$, and reveal it to the environment *Env*;
2. the environment picks a convex, G-Lipschitz loss function $l_t : \mathcal{X} \to \mathbb{R}$;
3. we suffer the loss $l_t(x_t)$, and observe a subgradient $g_t \in \partial l_t(x_t)$;
4. the environment determines if the game should stop – let T be the total number of rounds.

Performance metric: Regret

After the game ends, our total loss is $\sum_{t=1}^{T} l_t(x_t)$.

- We want to guarantee low total loss, but can we even do that?
- Let's say if all the losses l_t become $l_t + 1$, then the total loss is automatically increased by T.

A more sensible performance metric is a "comparative" one:

Definition. With an alternative sequence of decisions $u_1, \ldots, u_T$ called a comparator,

$$\mathrm{Regret}_T(Env, u_{1:T}) := \sum_{t=1}^{T} l_t(x_t) - \sum_{t=1}^{T} l_t(u_t).$$

Key idea. We minimize $\mathrm{Regret}_T(Env, u_{1:T})$ rather than the total loss.

Minimax online learning

However, $\mathrm{Regret}_T(Env, u_{1:T})$ itself is NOT a well-posed objective function yet...

- We don't know the behavior of Env throughout the game.
- Our "benchmark" $u_{1:T}$ should perform well on Env, so it's also unknown.

Classical workaround. Assume a class of Env and $u_{1:T}$, which defines the worst case regret

$$\max_{Env, u_{1:T}} \mathrm{Regret}_T(Env, u_{1:T}).$$

Minimax OL. Guarantee $\max_{Env, u_{1:T}} \mathrm{Regret}_T(Env, u_{1:T}) \leq o(T)$, for a class of Env and $u_{1:T}$.

- The backbone of "Robust + X", where X = optimization, control, ML,...

What does it really mean?

Dividing the regret definition by T,

$$\max_{Env, u_{1:T}} \left[\frac{1}{T} \sum_{t=1}^{T} l_t(x_t) - \frac{1}{T} \sum_{t=1}^{T} l_t(u_t) \right] \leq \frac{o(T)}{T} \xrightarrow{T \to \infty} 0.$$

Regardless of *Env*, we asymptotically perform no worse than any considered sequence $u_{1:T}$!

Or in other words,

- With more data, we eventually "learn" the class of $u_{1:T}$.
- On the generation of data (l_t), there's no statistical assumptions at all!
- Certainly, the result also applies to IID data.

Example: Empirical Risk Minimization (ERM)

The classical framework of machine learning consists of the following components:

1. Data Z, which is a r.v. following an unknown distribution $\mathcal{D}$;
2. Model parameter θ;
3. Loss function f, which maps the data and parameter into a real number.

NN Regression. (1) covariate-label pair; (2) parameters of NN; (3) NN structure + sq loss.

Goal. Minimizing the risk / generalization error $\mathbb{E}_{Z\sim\mathcal{D}}[f(\theta, Z)]$ over the parameter θ.

ERM. We only have a dataset $\bar{\mathcal{D}}$ sampled IID from $\mathcal{D}$. So, let's aim for the empirical risk,

$$\frac{1}{|\bar{\mathcal{D}}|}\sum_{Z_i\in\bar{\mathcal{D}}} f(\theta, Z_i).$$

Solving ERM by minimizing static regret

Typically, the dataset $\bar{\mathcal{D}}$ is so large that we have to process small pieces one at a time.

- Batch optimization $\Longrightarrow$ Online optimization!

OCO	Decision x_t	Loss l_t	Comparator u
ERM	The t-th iterate, θ_t	The t-th minibatch loss, $f(\cdot, Z_t)$	Empirical risk minimizer θ^*

- We don't care about the order within $\bar{\mathcal{D}} \Longrightarrow$ Static regret, $u_t = u$.

$$\mathrm{Regret}_T(Env, u) := \sum_{t=1}^{T} l_t(x_t) - \sum_{t=1}^{T} l_t(u).$$

An OCO algorithm with $o(T)$ static regret ensures, with large $|\bar{\mathcal{D}}|$,

$$\frac{1}{|\bar{\mathcal{D}}|}\sum_{i=1}^{|\bar{\mathcal{D}}|} f(\theta_i, Z_i) \leq \min_{\theta} \frac{1}{|\bar{\mathcal{D}}|}\sum_{Z_i \in \bar{\mathcal{D}}} f(\theta, Z_i) + o(1). \quad \text{"Successful" ERM!}$$

Part 2 – What is adaptive online learning?

In a nutshell

Let's only consider static regret and comparator adaptivity from this point.

Instead of aiming for the worst case regret bound (implicitly, it contains the complexity measure of the *Env* and *u* class)

$$\max_{Env,u} \mathrm{Regret}_T(Env, u) \leq o(T),$$

we seek a function $R(u, T)$ sublinear in T, such that

$$\max_{Env} \mathrm{Regret}_T(Env, u) \leq R(u, T).$$

Essentially: Problem complexity (of the *u* class) $\Longrightarrow$ instance complexity (of each *u*).

– Not just due to better analysis, we need better algorithms as well!

Online Gradient Descent: The default minimax algorithm

How do we achieve worst case regret bounds?

– We iterate with gradient feedback. $\Longrightarrow$ (Online) Gradient Descent.

OGD uses the projected gradient step $x_{t+1} = \Pi_{\mathcal{X}}(x_t - \eta g_t)$.

$$\sum_{t=1}^{T} l_t(x_t) - \sum_{t=1}^{T} l_t(u) \leq \frac{\|u - x_1\|^2}{2\eta} + \frac{G^2 T}{2}\eta.$$

– Let's assume the domain $\mathcal{X}$ is bounded with diameter D; a key assumption!

– Tuning $\eta = DG^{-1}T^{-1/2}$ guarantees the minimax optimal static regret

$$\sup_{Env;u\in\mathcal{X}} \mathrm{Regret}_T(Env, u) = O(DG\sqrt{T}).$$

– For any algorithm, there exists Env and u such that $\mathrm{Regret}_T(Env, u) \geq \Omega(DG\sqrt{T})$.

It is clear that the performance OGD depends on D, an overestimate of $\|u - x_1\|$.

- It's basically a scaling factor of the learning rate η!

What if D is known, but loose? Think about weather forecasting:

- The temperature is for sure within $[-1000°C, 1000°C]$. But can we set $D = 1000°C$?
- Predictions x_t are vulnerable to noise in g_t, due to large learning rate η.
- The regret bound is $O(DG\sqrt{T})$, which scales with loose D estimates.

What if D is even unknown, or ∞? No way to set η, and worse, $\max_u \mathrm{Regret}_T = \infty$!

Suppose we know $\|u - x_1\|$, we could have tuned η better, for a $O(\|u - x_1\| G\sqrt{T})$ bound.

- Realistically, this is impossible for OGD; but a very different algorithm can almost[1] do this!

Algorithmic interpretation

> *Although the characteristics of the problem instance ($\|u - x_1\|$) are unknown, we want to perform almost as if they are known at the beginning.*

Theoretical interpretation

> *Replacing problem complexity (D) in the minimax optimal bound by instance complexity ($\|u - x_1\|$), we aim to achieve near instance optimality.*

[1] "Almost" and "near": up to poly-logarithmic factors.

Concrete benefits of adaptivity

Generality

- Fewer restrictions allows handling harder settings, e.g., unbounded $\mathcal{X}$ ($D = \infty$).

Robustness to suboptimal hyperparameter tuning

- Logarithmic rather than polynomial dependence. "Parameter-free".

⭐ Incorporation of Bayesian priors

- x_1 can be "guessed", such that for good comparator u, $\|u - x_1\|$ is small.
- Intuition: the more prior knowledge we have on something, the easier it becomes!
- Bridges domain knowledge / scientific models with online data.

Part 3 – How do we achieve comparator adaptivity?

Comparator adaptivity on unconstrained domain

Assume $\mathcal{X}$ is unbounded $\implies$ The optimal tuning of OGD fails.

Mirroring the standard $O(DG\sqrt{T})$ bound of OGD, our goal is

$$\max_{Env} \mathrm{Regret}_T(Env, u) = \tilde{O}(\|u - x_1\| \, G\sqrt{T}),$$

where $\tilde{O}$ hides poly-logarithmic factors.

Prior work

It is known that

- the problem in $\mathbb{R}^d$ can be reduced to $\mathbb{R}$ [Cutkosky and Orabona, '18];
- given a suitable potential function $V(t, S)$, the resulting 1D problem can be solved by the potential framework [McMahan and Orabona, '14; Orabona and Pal, '16; Mhammedi and Koolen, '20]

$$x_t = \nabla_S V\left(t, -G^{-1}\sum_{i=1}^{t-1} g_i\right).$$

Limitation

- Designing good V relies on guessing, so the best known V is still suboptimal in certain sense.

Can we make the design of V easier and quantitatively stronger?

Key idea: Designing V in continuous time

Why is guessing hard? The search space for the function V is too large.

A key result when T is given:

- Cover ['65] showed that all achievable regret bounds can be achieved by computing V through dynamic programming.
- For OCO, starting from a terminal potential $V(T, \cdot)$ satisfying certain "achievability" condition,

$$V(t, S) = \min_{x \in \mathbb{R}} \max_{g \in [-1,1]} \left[V(t+1, S-g) + gx \right], \quad \forall t \leq T-1; \forall S \in \mathbb{R}.$$

(*) Intuitively, "achievability" is similar to a no-free-lunch theorem: if a regret bound doesn't imply making net profit on a purely random market, then it's achievable.

Without knowing T, there is no "terminal" anymore.

Key idea: Designing V in continuous time

Can we still properly reduce the search space?

Continuous time approximation: scaling time steps and data

- Bellman equation $\Rightarrow$ Backward Heat Equation

$$\nabla_t V + \frac{1}{2}\nabla_{SS} V = 0.$$

- No boundary condition $\Longrightarrow$ we have a solution class.
- Searching in the solution class is a lot more structured task!
- Building on [Drenska and Kohn, '20; Harvey et al., '21], but with a more algorithmic focus, on adaptive OL.

A new potential for unconstrained OCO. Hard to find without going through CT.

$$V(t,S)=C\sqrt{t}\left[2\int_0^{\frac{S}{\sqrt{2t}}}\left(\int_0^u \exp(x^2)dx\right)du-1\right].$$

Then, verification argument, which is most of the heavy lifting.

SOTA regret bound. Given any $C>0$, for all $T\in\mathbb{N}_+$ and $u\in\mathbb{R}$,

$$\max_{Env}\mathrm{Regret}_T(Env,u)\le C\sqrt{T}+|u-x_1|\sqrt{2T}\left[\sqrt{\log\left(1+\frac{|u-x_1|}{\sqrt{2}C}\right)}+2\right].$$

- The first "practical" algorithm that achieves the optimal $O(\sqrt{T})$ rate.
- Optimal leading constant $\sqrt{2}$.
- Extension: simultaneous adaptivity to both *Env* and *u* [arXiv'23].

Extension: OCO with switching cost

Suppose that besides the loss $l_t(x_t)$, we also suffer a switching cost $\lambda \|x_t - x_{t-1}\|$.

Performance metric. In 1D, the augmented regret

$$\mathrm{Regret}_T^{\lambda}(Env, u) := \sum_{t=1}^{T} g_t(x_t - u) + \lambda \sum_{t=1}^{T-1} |x_{t+1} - x_t| .$$

Motivation. Smooth operation, and OL with long term effect [Agarwal et al., '19].

Challenge. The right amount of optimism.

We [AISTATS'22] proposed the first comparator adaptive algorithm for this setting, improved in [NeurIPS'22] through CT.

Continuous time reveals algorithmic connections [NeurIPS'22]

Discrete time recursion.

$$V(t-1,S)=\max_{g\in[-1,1]}\left\{V(t,S-g)+g\nabla_S V(t,S)+\lambda\left|\nabla_S V(t,S)-\nabla_S V(t+1,S-g)\right|\right\}.$$

Continuous time approximation. Still BHE, but with a different diffusivity constant, $\frac{1}{2}+\lambda$.

Algorithmic interpretation

- Change of variable $\Rightarrow$ dual space scaling
- CT reveals internal connections among different OL settings.

Takeaway

Online learning is a type of ML where data is sequentially revealed and possibly interactive.

Adaptive OL improves standard minimax OL by achieving instance optimality, rather than worst case optimality.

PDE simplifies the design of adaptive OL algorithms by providing a class of potential functions that approximately satisfy the minimax Bellman equation.