Minimax Adaptive Boosting in Online Non-Parametric Regression

Séminaire des doctorants du LPSM, Jussieu, Paris

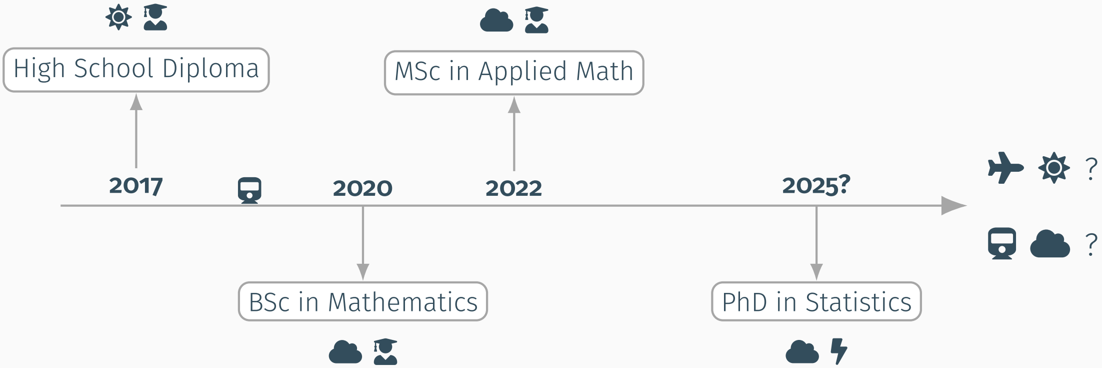
Paul Liautaud

December 9, 2024

Sorbonne University, Paris

Welcome message & academic journey

Happy to welcome M2 students for this session!



Joint work with

Pierre Gaillard
CR Inria/UGA

Olivier Wintenberger
PR LPSM/SU

Online Learning &
Non-Parametric Regression

Classical Machine Learning

The learner:

❶ observes a **whole training dataset** with labels/targets:

$$(x_1, y_1), \ldots, (x_T, y_T) \overset{\text{iid}}{\sim} (X, Y) \text{ with distribution } \mathbb{P} \text{ over } \mathcal{X} \times \mathcal{Y}.$$

Classical Machine Learning

The learner:

$\longrightarrow$ Learning method

❶ observes a **whole training dataset** with labels/targets:

$$(x_1, y_1), \ldots, (x_T, y_T) \overset{\text{iid}}{\sim} (X, Y) \text{ with distribution } \mathbb{P} \text{ over } \mathcal{X} \times \mathcal{Y}.$$

❷ builds a function $\hat{f} : \mathcal{X} \to \mathcal{Y} \in \mathcal{F}$ with small risk $\mathbb{E}_{\mathbb{P}}[\ell(\hat{f}(X), Y)]$ by minimizing:

$$R(\hat{f}) = \frac{1}{T} \sum_{t=1}^{T} \ell(\hat{f}(x_t), y_t),$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a prescribed loss function.

Classical Machine Learning

The learner: 

$\longrightarrow$ Learning method $\longrightarrow$ Prediction on test data

❶ observes a **whole training dataset** with labels/targets:

$$(x_1, y_1), \ldots, (x_T, y_T) \overset{\text{iid}}{\sim} (X, Y) \text{ with distribution } \mathbb{P} \text{ over } \mathcal{X} \times \mathcal{Y}.$$

❷ builds a function $\hat{f} : \mathcal{X} \to \mathcal{Y} \in \mathcal{F}$ with small risk $\mathbb{E}_{\mathbb{P}}[\ell(\hat{f}(X), Y)]$ by minimizing:

$$R(\hat{f}) = \frac{1}{T} \sum_{t=1}^{T} \ell(\hat{f}(x_t), y_t),$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a prescribed loss function.

❸ controls the error of new data if they are similar to the training data.

Classical Machine Learning

The learner:  $\longrightarrow$ Learning method $\longrightarrow$ Prediction on test data

❶ observes a **whole training dataset** with labels/targets:

$$(x_1, y_1), \ldots, (x_T, y_T) \overset{\text{iid}}{\sim} (X, Y) \text{ with distribution } \mathbb{P} \text{ over } \mathcal{X} \times \mathcal{Y}.$$

❷ builds a function $\hat{f} : \mathcal{X} \to \mathcal{Y} \in \mathcal{F}$ with small risk $\mathbb{E}_{\mathbb{P}}[\ell(\hat{f}(X), Y)]$ by minimizing:

$$R(\hat{f}) = \frac{1}{T} \sum_{t=1}^{T} \ell(\hat{f}(x_t), y_t),$$

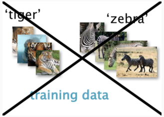
where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a prescribed loss function.

❸ controls the error of new data if they are similar to the training data.

✗ We won't deal with it! ✗

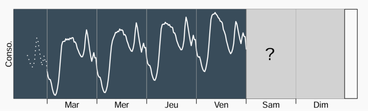
In **sequential learning**:

- data are acquired and treated **on the fly**;
- data are **not** necessarily **iid**, possibly **adversarial**;
- feedbacks are received and algorithms updated **step by step**.



Why **online** learning? In some applications, the environment may **evolve over time** and data may be available **sequentially**, e.g.:

- ads to display,
- electricity consumption forecast,
- spam detection,
- aggregation of expert knowledge.

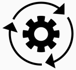


Setting of the talk (1/2)

The scenario is as follows:

At each round $t = 1, \dots, T$, the learner or algorithm

1. observes input $x_t \in \mathcal{X}$
2. makes prediction $\hat{f}_t(x_t) \in \mathbb{R}$ — Choose $\hat{f}_t$ before observing ℓ_t
3. incurs loss $\ell_t(\hat{f}_t(x_t))$ and discover gradients g_t — No assumptions on how ℓ_t is generated!
4. updates prediction function $\hat{f}_t \to \hat{f}_{t+1}$

$\dots \to x_t =$  $\to$  $\to$ 'zebra' $\to$  $\to x_{t+1} =$  $\to \dots$

Assumptions:

- $\ell_1, \dots, \ell_T$ are convex, differentiable and G-Lipschitz, with $G > 0$;
- $\mathcal{X} \subset \mathbb{R}^d$ bounded subset with $|\mathcal{X}| = \sup_{x,x' \in \mathcal{X}} \|x - x'\|_\infty$.

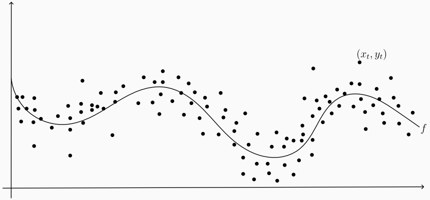
Setting of the talk (2/2)

Goal: find $\hat{f}_1, \ldots, \hat{f}_T$ that...

minimize the cumulative loss $\Leftrightarrow$ predict almost as well as the best function f

$$\sum_{t=1}^{T} \ell_t(\hat{f}_t(x_t)) \qquad \underbrace{\sum_{t=1}^{T} \ell_t(\hat{f}_t(x_t)) - \sum_{t=1}^{T} \ell_t(f(x_t))}_{:=\mathrm{Reg}_T(f)}$$

'Non-Parametric regression': $(\hat{f}_t)$ is compared to benchmark functions $f \in \mathcal{F}$, e.g. Lipschitz



Regret analysis

- We want $\hat{f}_1, \dots, \hat{f}_T$ such that regret against $f \in \mathcal{F}$ is as **small** as possible

$$\mathrm{Reg}_T(f) = \underbrace{\sum_{t=1}^{T} \ell_t(\hat{f}_t(x_t))}_{\text{our performance}} \quad - \quad \underbrace{\sum_{t=1}^{T} \ell_t(f(x_t))}_{\text{reference performance}} \quad \underbrace{= o(T)}_{\text{goal}}$$

⚠ **Difficulty: no stochastic assumption** on data (x_t, ℓ_t)!

- $\hat{f}_1, \dots, \hat{f}_T$ have to perform **well** with all **arbitrary** time series i.e. approaching

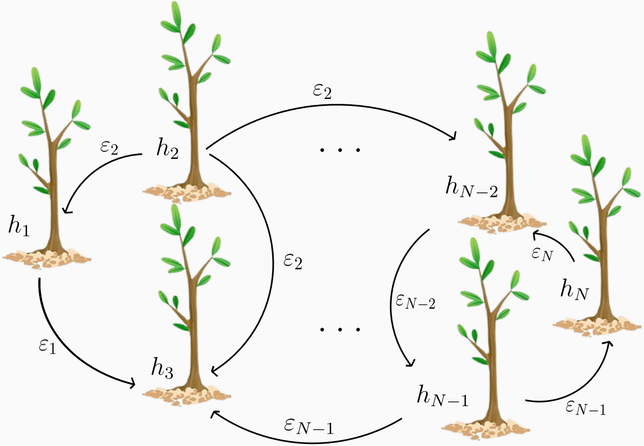
$$\inf_{\hat{f}_1} \sup_{x_1,\ell_1} \inf_{\hat{f}_2} \sup_{x_2,\ell_2} \cdots \inf_{\hat{f}_T} \sup_{x_T,\ell_T} \sup_{f \in \mathcal{F}} \mathrm{Reg}_T(f).$$

? How to sequentially build such predictors?

Building Predictions with Online Gradient Boosting

Boosting uses ”wisdom of the crowd”

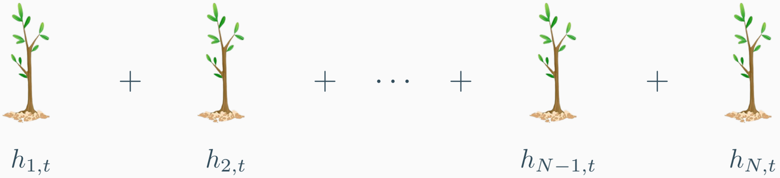
- **Boosting**: ensemble method combining multiple **weak learners** to create a **strong learner**
- Each **model** corrects/learns from errors of its peers
- Results in a **highly accurate** predictive model [1]



[1] e.g. AdaBoost and XGBoost

How to deal with weak learners?

- $\mathcal{W} \subset \mathbb{R}^{\mathcal{X}}$ a set of real valued functions $\mathcal{X} \to \mathbb{R}$ (e.g. trees, piecewise constant functions)
- $\mathrm{span}_N(\mathcal{W}) := \{\sum_{n=1}^{N} \beta_n h_n,\ h_n \in \mathcal{W}, \beta_n \in \mathbb{R}\}$ linear function space associated to $\mathcal{W}$
- For each $t = 1, \ldots, T$, we use and train $N \geqslant 1$ **sequential predictors** from $\mathcal{W}$



and we form a **strong predictor** in $\mathrm{span}_N(\mathcal{W})$, at any time $t \geqslant 1$, as

$$\hat{f}_t = \sum_{n=1}^{N} \beta_{n,t} h_{n,t}\,, \qquad \beta_{n,t} \in \mathbb{R}, n \in [N]$$

A new Online Gradient Boosting procedure

Goal: find a sequence of functions

$$\hat{f}_t = \sum_n \beta_{n,t} h_{n,t} \in \mathrm{span}_N(\mathcal{W}), \qquad 1 \leqslant t \leqslant T,$$

At $t \geqslant 1$, each $n \in [N]$ is boosted with **OGB** as:

1. Predict $\hat{f}_t(x_t)$, define $\hat{f}_{-n,t} = \hat{f}_t - \beta_{n,t} h_{n,t}$
2. $(\beta_{n,t}, h_{n,t})$ receives its gradient

$$g_{n,t} = \left[\nabla_{(\beta_n, h_n)} \ell_t\left(\hat{f}_{-n,t}(x_t) + \beta_n h_n(x_t)\right)\right]_{(\beta_n,h_n)=(\beta_{n,t},h_{n,t})}$$

3. Find $(\beta_{n,t+1}, h_{n,t+1}) \in \mathbb{R} \times \mathcal{W}$ to solve

$$\min_{\beta_n, h_n} \ell_t(\hat{f}_{-n,t}(x_t) + \beta_n h_n(x_t)) \qquad (1)$$

using gradient $g_{n,t}$.

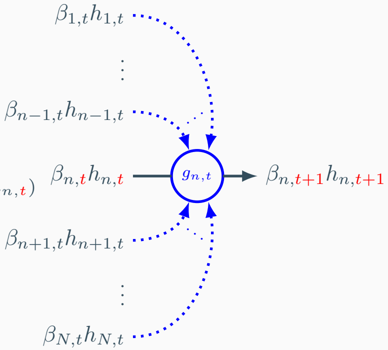


Figure 1: Boosting at time t.

Online Gradient Boosting in Chaining-Tree

Tree-based method

🌲 A **regular decision-tree** $(\mathcal{T}, \bar{\mathcal{X}}, \bar{\mathcal{W}})$ over $\mathcal{X}$ is made of:

- a set of nodes $\mathcal{N}(\mathcal{T})$ including leaves $\mathcal{L}(\mathcal{T})$;
- a family of subregions

$$\bar{\mathcal{X}} = \{\mathcal{X}_n, n \in \mathcal{N}(\mathcal{T})\}$$

  partitionning $\mathcal{X}$ by level;
- a family of prediction functions

$$\bar{\mathcal{W}} = \{h_n, n \in \mathcal{N}(\mathcal{T})\}.$$

💡 The idea will be to boost the predictive nodes $\bar{\mathcal{W}}$.

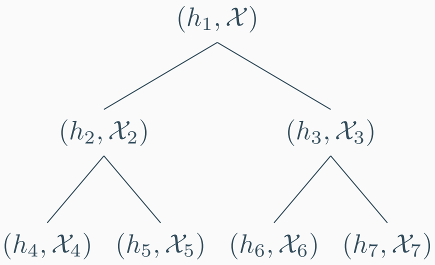


Figure 2: Example of $\mathcal{T}$ with depth $\mathrm{d}(\mathcal{T}) = 3$ over $\mathcal{X} \subset \mathbb{R}$.

Chaining-Tree

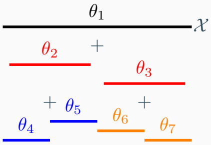


Figure 3: Prediction of a CT $\mathcal{T}$ of depth $\mathrm{d}(\mathcal{T}) = 3$ on $\mathcal{X} \subset \mathbb{R}$.

Definition (Chaining-Tree)

A Chaining-Tree (CT) prediction function $\hat{f}$ over $\mathcal{X}$ is defined as

$$\hat{f}(x) = \sum_{n \in \mathcal{N}(\mathcal{T})} h_n(x)\,, \quad x \in \mathcal{X},$$

where:

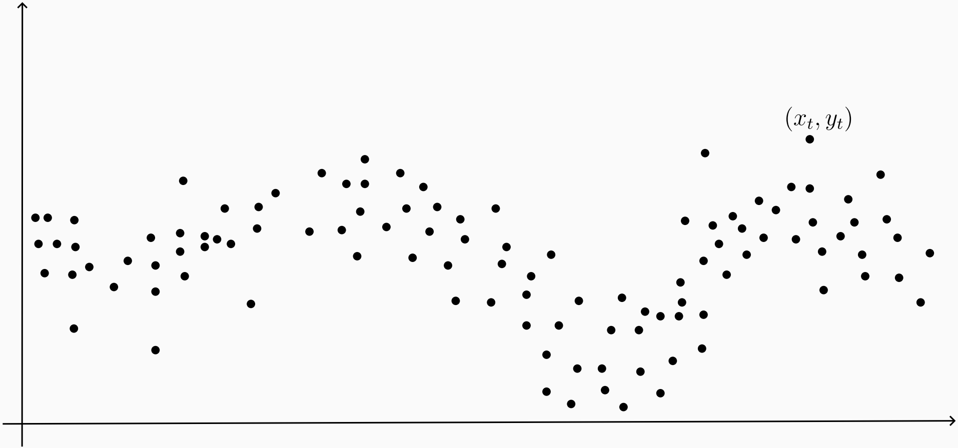
- $h_n(x) = \theta_n \mathbb{1}_{x \in \mathcal{X}_n}$ are constant functions;
- each interior node $n \in \mathcal{N}(\mathcal{T}) \backslash \mathcal{L}(\mathcal{T})$ has 2^d children forming a regular partition of $\mathcal{X}_n$.

<u>Remark:</u> contrary to standard methods, we predict with **all** nodes $n \in \mathcal{N}(\mathcal{T})$.

Approximation process by a Chaining-Tree

- Assume $\ell_t(\hat{y}) = (\hat{y} - y_t)^2$ and we launch a CT $\mathcal{T}$ with depth $\mathrm{d}(\mathcal{T}) = 1, 2, 3$, over T data.

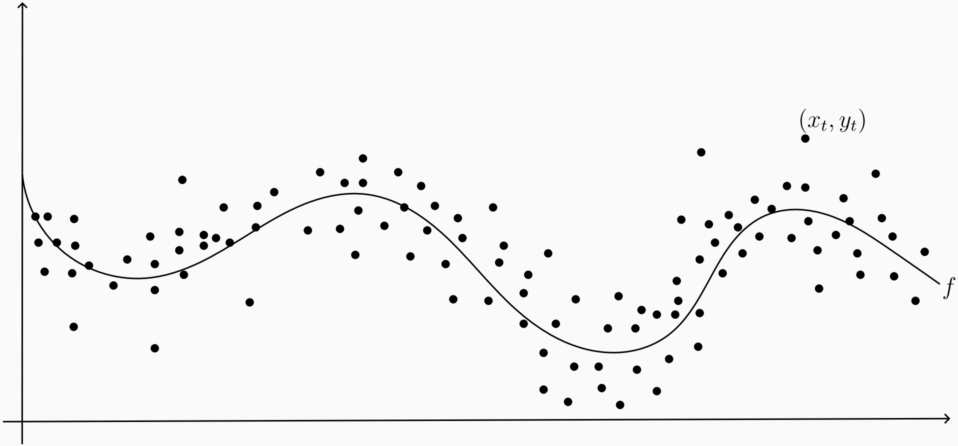
We have the following illustration:



Approximation process by a Chaining-Tree

- Assume $\ell_t(\hat{y}) = (\hat{y} - y_t)^2$ and we launch a CT $\mathcal{T}$ with depth $\mathrm{d}(\mathcal{T}) = 1, 2, 3$, over T data.

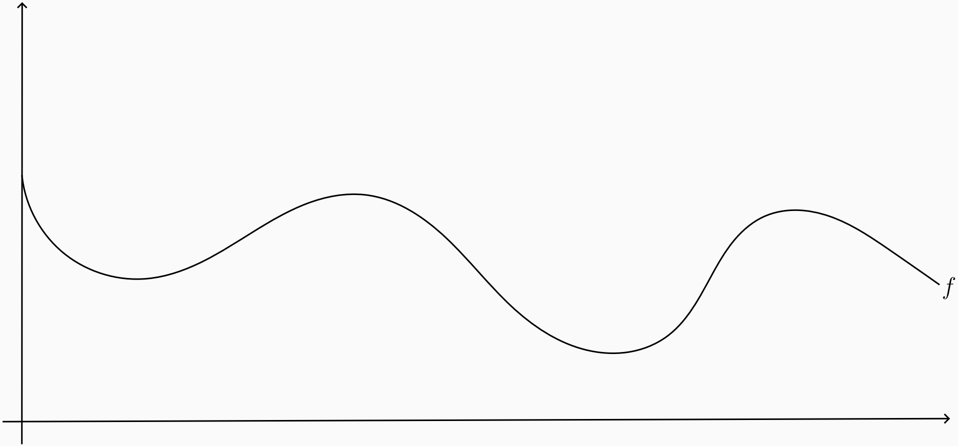
We have the following illustration:



Approximation process by a Chaining-Tree

- Assume $\ell_t(\hat{y}) = (\hat{y} - y_t)^2$ and we launch a CT $\mathcal{T}$ with depth $\mathrm{d}(\mathcal{T}) = 1, 2, 3$, over T data.

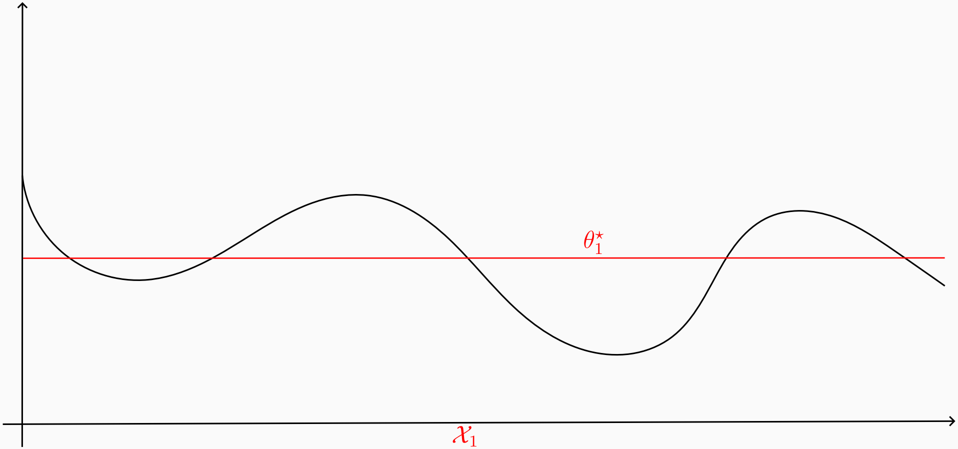
We have the following illustration:



Approximation process by a Chaining-Tree

- Assume $\ell_t(\hat{y}) = (\hat{y} - y_t)^2$ and we launch a CT $\mathcal{T}$ with depth $\mathrm{d}(\mathcal{T}) = 1, 2, 3$, over T data.

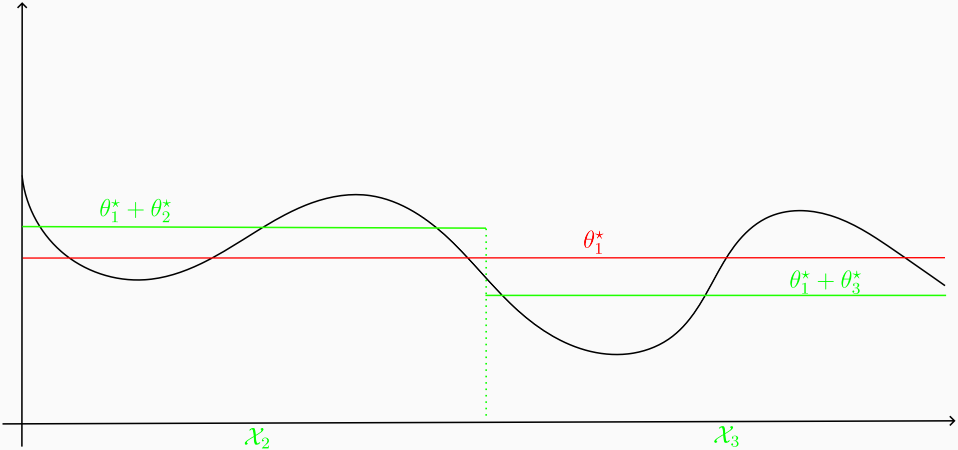
We have the following illustration:



Approximation process by a Chaining-Tree

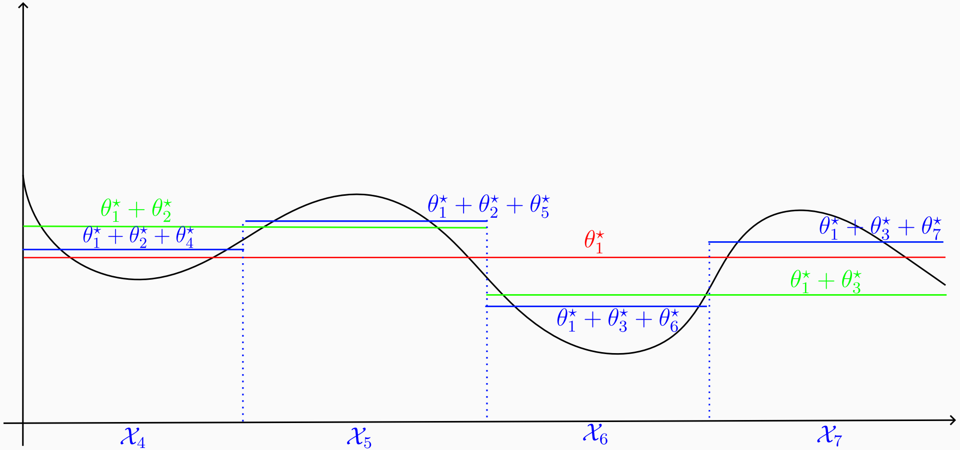
- Assume $\ell_t(\hat{y}) = (\hat{y} - y_t)^2$ and we launch a CT $\mathcal{T}$ with depth $\mathrm{d}(\mathcal{T}) = 1, 2, 3$, over T data.

We have the following illustration:



Approximation process by a Chaining-Tree

- Assume $\ell_t(\hat{y}) = (\hat{y} - y_t)^2$ and we launch a CT $\mathcal{T}$ with depth $\mathrm{d}(\mathcal{T}) = 1, 2, 3$, over T data. We have the following illustration:



Online Boosting in a Chaining-Tree

Goal: Sequentially training CT $\mathcal{T}$, i.e. tuning over time the family

$$\bar{\mathcal{W}}_t := \{h_{n,t} = \theta_{n,t}\mathbb{1}_{\mathcal{X}_n}, n \in \mathcal{N}(\mathcal{T})\}.$$

We use OGB on $\bar{\mathcal{W}}_t$, with $\beta_n = 1$, $N = |\mathcal{N}(\mathcal{T})|$, $g_{n,t} = \ell_t'(\hat{f}_t(x_t))\mathbb{1}_{x_t \in \mathcal{X}_n}$ and a parameter-free [2] procedure in minimization step ❸, i.e. for any node $n \in \mathcal{N}(\mathcal{T})$,

$$\textstyle\sum_{t \in T_n} g_{n,t}(\theta_{n,t} - \theta_n) \lesssim G|\theta_n|\sqrt{|T_n|}\,, \qquad \text{with} \qquad \theta_n \in \mathbb{R}, T_n = \{1 \leqslant t \leqslant T, g_{n,t} \neq 0\}\,,$$

Target class $\mathcal{F}$: α-Hölder continous functions $(0 < \alpha \leqslant 1)$ over $\mathcal{X} \subset \mathbb{R}^d$:

$$\mathrm{Lip}_L^\alpha(\mathcal{X}) := \{f : \mathcal{X} \to \mathbb{R} : |f(x) - f(x')| \leqslant L\|x - x'\|_\infty^\alpha\,, \forall x, x' \in \mathcal{X}\}.$$

[2] Orabona and Pál, "Coin betting and parameter-free online learning"; Mhammedi and Koolen, "Lipschitz and comparator-norm adaptivity in online learning"; Cutkosky and Orabona, "Black-box reductions for parameter-free online learning in banach spaces".

Optimal regret against Lipschitz functions

Theorem (Regret of OGB-CT vs Lipschitz functions - Liautaud et al. (2024))

OGB on CT $(\mathcal{T}, \bar{\mathcal{X}}, \bar{\mathcal{W}})$ *with* $\mathcal{X}_{\mathrm{root}} = \mathcal{X}$, $\theta_{n,1} = 0, n \in \mathcal{N}(\mathcal{T})$ *and* $\mathrm{d}(\mathcal{T}) = \frac{1}{d}\log_2 T$ *has regret:*

$$\sup_{f \in \mathrm{Lip}_L^\alpha(\mathcal{X})} \mathrm{Reg}_T(f) \lesssim GLX^\alpha \begin{cases} \sqrt{T} & \text{if } d < 2\alpha\,, \\ \log_2 T\sqrt{T} & \text{if } d = 2\alpha\,, \\ T^{1-\frac{\alpha}{d}} & \text{if } d > 2\alpha\,, \end{cases}$$

for any $L > 0, \alpha \in (0,1]$.

Our rates are **minimax** over Lip_L^α (Rakhlin et al. (2015)) + we **do not need** prior knowledge of neither L nor α.

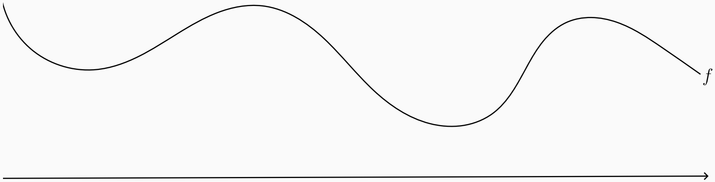
Computationally tractable: x_t falls into only one subregion $\mathcal{X}_n$ for each level $1, \ldots, \mathrm{d}(\mathcal{T})$: we update $\mathcal{O}(\frac{1}{d}\log_2(T))$ at each round.

Adaptive Boosting in Online NonParametric Regression

Adaptivity to local profile of the competitor

Goal: learn the best pruned tree i.e. the best partition over $\mathcal{X}$ to fit the competitor.

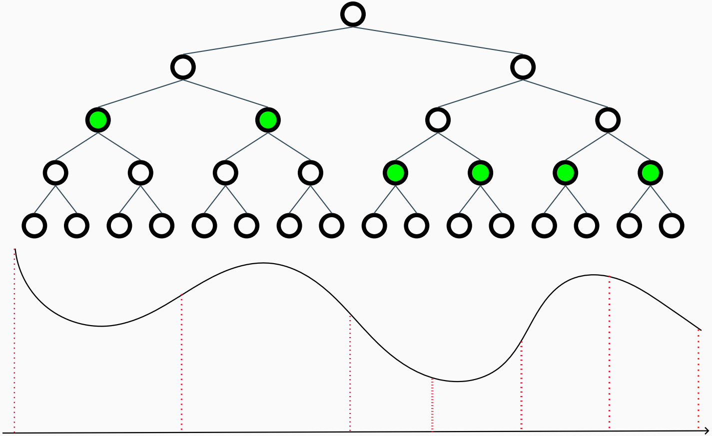
Example 1:



Adaptivity to local profile of the competitor

Goal: learn the best pruned tree i.e. the best partition over $\mathcal{X}$ to fit the competitor.

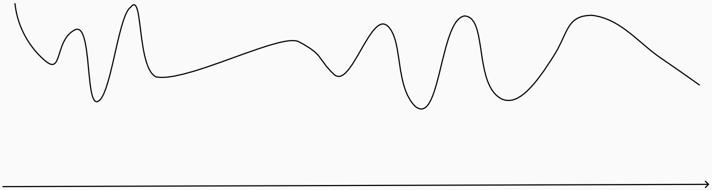
Example 1:



Adaptivity to local profile of the competitor

Goal: learn the best pruned tree i.e. the best partition over $\mathcal{X}$ to fit the competitor.

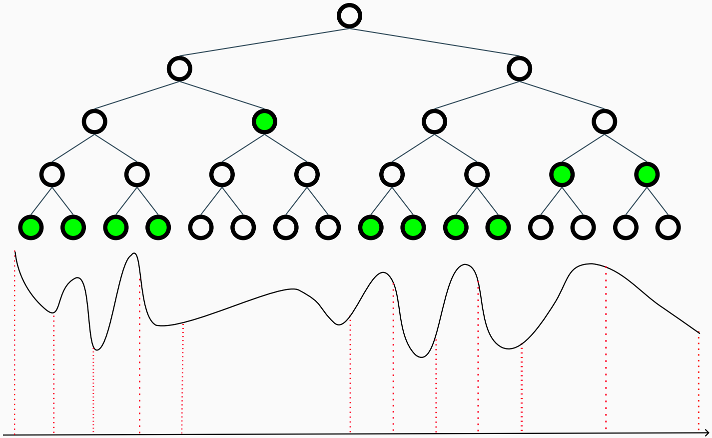
Example 2:



Adaptivity to local profile of the competitor

Goal: learn the best pruned tree i.e. the best partition over $\mathcal{X}$ to fit the competitor.

Example 2:



Locally Adaptive Boosting - LocAdaBoost

We base our predictions on a core tree $\mathcal{T}_0$ associated to:

$$\hat{f}_t(x_t) = \sum_{n \in \mathcal{N}(\mathcal{T}_0)} w_{n,t} \hat{f}_{n,t}(x_t)\,, \quad \forall t \geqslant 1,$$

where for any $n \in \mathcal{N}(\mathcal{T}_0)$:

- $\hat{f}_n$ is a CT rooted at $\mathcal{X}_n$ and trained as before;
- $w_{n,t}$ weight associated.

We use OGB on

- $\beta_{n,t} = w_{n,t}, n \in \mathcal{N}(\mathcal{T}_0)$;
- and gradient $\tilde{\mathbf{g}}_t = \nabla_{(w_n)} \ell_t(\hat{f}_t(x_t))|_{(w_n)=(w_{n,t})}$.

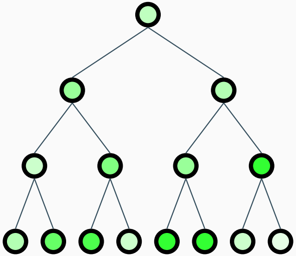


Figure 4: $\mathcal{T}_0$

Optimal and Locally Adaptive Regret (1/2)

Theorem (Locally Adaptive Regret, case $d = 1, \alpha > \frac{1}{2}$ - Liautaud et al. (2024))

Under assumptions, for any $f \in \mathrm{Lip}_L^\alpha(\mathcal{X})$, `LocAdaBoost` *achieves*

$$\mathrm{Reg}_T(f) \lesssim \inf_{\mathcal{T} \in \mathcal{P}(\mathcal{T}_0)} \left\{ \sqrt{T|\mathcal{L}(\mathcal{T})|} + |\mathcal{L}(\mathcal{T})| + |\mathcal{X}|^\alpha \sum_{n \in \mathcal{L}(\mathcal{T})} L_n(f) 2^{-\alpha \mathrm{d}(n)} \sqrt{|T_n|} \right\},$$

with $L_n(f)$ local Hölder constants.

If (ℓ_t) are exp-concave [3]

$$\mathrm{Reg}_T(f) \lesssim \inf_{\mathcal{T} \in \mathcal{P}(\mathcal{T}_0)} \left\{ |\mathcal{L}(\mathcal{T})| + |\mathcal{X}|^\alpha \sum_{n \in \mathcal{L}(\mathcal{T})} L_n(f) 2^{-\alpha \mathrm{d}(n)} \sqrt{|T_n|} \right\}$$

Optimal and Locally Adaptive Regret (2/2)

Corollary (Minimax Regret, $d = 1, \alpha > \frac{1}{2}$ - Liautaud et al. (2024))

For any $f \in \mathrm{Lip}_L^\alpha(\mathcal{X}), L > 0$, `LocAdaBoost` achieves

$$\mathrm{Reg}_T(f) \lesssim \begin{cases} (|\mathcal{X}|^\alpha \bar{L}(f))^{\frac{2}{2\alpha+1}} T^{\frac{1}{2\alpha+1}} & \text{if } \ell_t \text{ are exp-concave}, \\ (|\mathcal{X}|^\alpha \bar{L}(f))^{\frac{1}{2\alpha}} \sqrt{T}, \end{cases}$$

where $\bar{L}(f) = \left(\frac{1}{|\mathcal{X}|} \sum_{n \in \mathcal{L}(\mathcal{T})} |\mathcal{X}_n| L_n(f)^{1/\alpha}\right)^\alpha$.

- Remark: it could also adapt to local regularities (α_n)
- ✔ Minimax optimality
- ✔ Adaptivity to local regularities (L_n) and α
- ✔ Adaptivity to the loss curvature

[3] e.g. squared, logistic loss

Conclusion

- New generic Online Gradient Boosting procedure;
- Online Gradient Boosting coupled with Chaining-Tree achieve **minimax regret**;
- Our unique `LocAdaBoost` algorithm both adapts optimaly to **local regularities** of the competitor and **curvature** of sequential losses;
- **First** constructive algorithm to achieve **optimal locally adaptive regret**;
- Future work: extend the boosting procedure to other learners to approach other classes of functions.

Thank you!

Questions?

[3] Link to the paper: `https://arxiv.org/abs/2410.03363`

Comparison with the litterature

Ref.	Assumptions	Upper bound
[4]	(ℓ_t) exp-concave, $L > 0$ unknown	$\min\left\{\sqrt{LT}, L^{\frac{2}{3}}T^{\frac{1}{3}}\right\}$
	(ℓ_t) convex, $L > 0$ unknown	$\sqrt{LT}$
[5]	(ℓ_t) square loss, $L > 0$ unknown	$\sqrt{LT}$
[6]	(ℓ_t) absolute loss, $L > 0$ known	$L^{\frac{1}{3}}T^{\frac{2}{3}}$
	(ℓ_t) square loss, $L > 0$ known	$\sqrt{LT}$
[7]	(ℓ_t) square loss, $L = 1$ known	$T^{\frac{1}{3}}$
[8]	(ℓ_t) convex, $L = 1$ known	$\sqrt{T}$

[4] Liautaud, Gaillard, and Wintenberger, "Minimax Adaptive Boosting for Online Nonparametric Regression".
[5] Kuzborskij and Cesa-Bianchi, "Locally-adaptive nonparametric online learning".
[6] Hazan, Agarwal, and Kale, "Logarithmic regret algorithms for online convex optimization".
[7] Gaillard and Gerchinovitz, "A Chaining Algorithm for Online Nonparametric Regression".
[8] Cesa-Bianchi et al., "Algorithmic chaining and the role of partial feedback in online nonparametric learning".

What about the excess-risk in batch learning?

- **Online regret bound** againt any $f \in \mathcal{F}$:

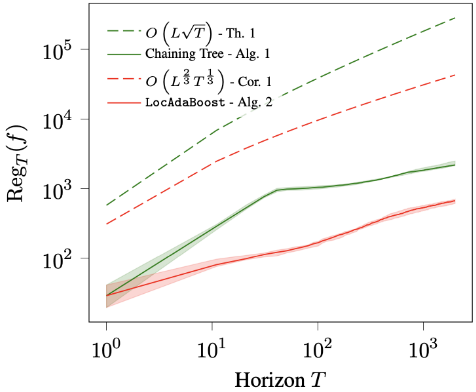
$$\frac{1}{T}\operatorname{Reg}_T(f) = \frac{1}{T}\sum_{t=1}^{T}(\hat{f}_t(x_t) - y_t)^2 - \frac{1}{T}\sum_{t=1}^{T}(f(x_t) - y_t)^2 = o(1).$$

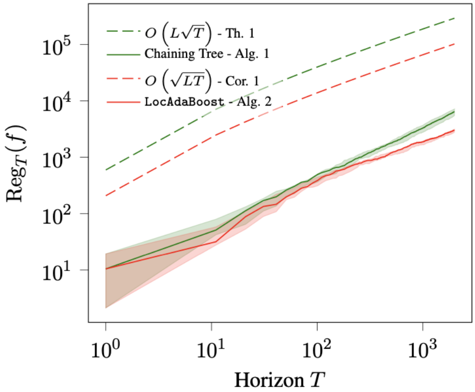
- If $\{(x_t, y_t)\}_{t=1}^{T} \overset{\text{iid}}{\sim} (X, Y), \ell_t(\hat{y}) = (\hat{y} - y_t)^2$, excess risk of $\bar{f}_T = \frac{1}{T}\sum_{t=1}^{T}\hat{f}_t$ is bounded as

$$\begin{aligned}
\mathbb{E}\big[(\bar{f}_T(X) - Y)^2\big] - \mathbb{E}\big[(f(X) - Y)^2\big] &\overset{\text{Convexity}}{\leqslant} \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[(\hat{f}_t(X) - Y)^2\big] - \mathbb{E}\big[(f(X) - Y)^2\big] \\
&= \frac{1}{T}\mathbb{E}\big[\operatorname{Reg}_T(f)\big] = o(1).
\end{aligned}$$

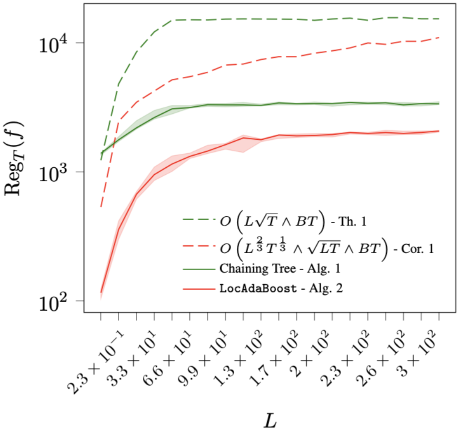
Experiments (1/3)

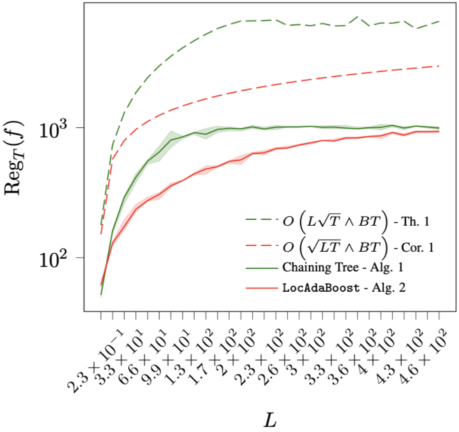
Regression setting: $y_t = f(x_t) + \varepsilon_t$, where $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$ with $\sigma = 0.5, f(x) = \sin(10x) + \cos(5x) + 5$, for $x \in \mathcal{X} = [0, 1]$ and $\sup_x |f'(x)| \leqslant 15 =: L$.



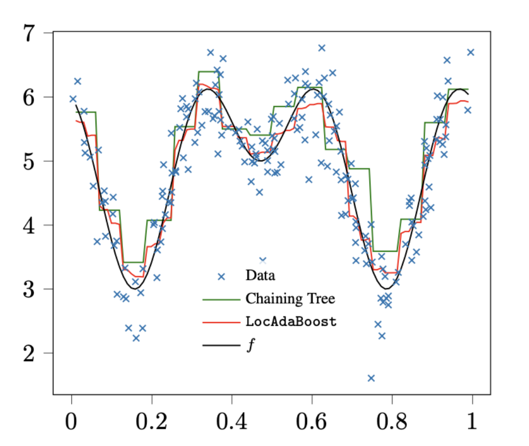


Experiments (2/3)





Experiments (3/3)



References

Cesa-Bianchi, Nicolò and Gábor Lugosi (2006). *Prediction, Learning, and Games*. Cambridge University Press.

Cesa-Bianchi, Nicolò et al. (2017). “Algorithmic chaining and the role of partial feedback in online nonparametric learning”. In: *Conference on Learning Theory*. PMLR, pp. 465–481.

Cutkosky, Ashok and Francesco Orabona (2018). “Black-box reductions for parameter-free online learning in banach spaces”. In: *Conference On Learning Theory*. PMLR, pp. 1493–1529.

Gaillard, Pierre and Sebastien Gerchinovitz (2015). “A Chaining Algorithm for Online Nonparametric Regression”. In: *COLT*.

Hazan, Elad, Amit Agarwal, and Satyen Kale (2007). “Logarithmic regret algorithms for online convex optimization”. In: *Machine Learning* 69.2, pp. 169–192.

Kuzborskij, Ilja and Nicolo Cesa-Bianchi (2020). “Locally-adaptive nonparametric online learning”. In: *Advances in Neural Information Processing Systems* 33, pp. 1679–1689.

Liautaud, Paul, Pierre Gaillard, and Olivier Wintenberger (2024). “Minimax Adaptive Boosting for Online Nonparametric Regression”. In: *arXiv preprint arXiv:2410.03363*.

Mhammedi, Zakaria and Wouter M Koolen (2020). “Lipschitz and comparator-norm adaptivity in online learning”. In: *Conference on Learning Theory*. PMLR, pp. 2858–2887.

Orabona, Francesco and Dávid Pál (2016). “Coin betting and parameter-free online learning”. In: *Advances in Neural Information Processing Systems* 29.

Rakhlin, Alexander and Karthik Sridharan (2014). “Online non-parametric regression”. In: *Conference on Learning Theory*. PMLR, pp. 1232–1264.

— (2015). “Online nonparametric regression with general loss functions”. In: *arXiv preprint arXiv:1501.06598*.

Zinkevich, Martin (2003). “Online convex programming and generalized infinitesimal gradient ascent”. In: *Proceedings of the 20th international conference on machine learning (icml-03)*, pp. 928–936.