Boosting in Online Non-Parametric Regression

Paul Liautaud

April 2, 2024

Sorbonne University, Paris

Joint work with

Pierre Gaillard
CR Inria/UGA

Olivier Wintenberger
PR Sorbonne University

Table of Contents

Online Learning &
Non-Parametric Regression

Classical Machine Learning

The learner

1. observes a **whole training dataset** with labels/targets,
2. builds a program to minimize the training error,
3. controls the error of new data if they are similar to the training data

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

Classical Machine Learning

The learner

1. observes a **whole training dataset** with labels/targets,
2. builds a program to minimize the training error,
3. controls the error of new data if they are similar to the training data

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

We won't deal with it!

A dive into Sequential Learning

Why **Online** Learning?

A dive into Sequential Learning

Why **Online** Learning? In some applications, the environment may evolve over time and the data may be available sequentially.
Examples:

- ads to display,
- electricity consumption forecast,
- spam detection,
- aggregation of experts/algorithms.

A dive into Sequential Learning

Why **Online** Learning? In some applications, the environment may evolve over time and the data may be available sequentially.
Examples:

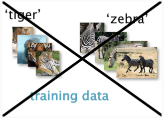
- ads to display,
- electricity consumption forecast,
- spam detection,
- aggregation of experts/algorithms.

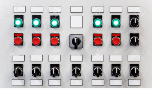
We need Online/Sequential Learning!

A dive into Sequential Learning

In **sequential learning**:

- Data are acquired and treated on the fly,
- Feedbacks are received and algorithms updated step by step.



 →  → zebra → Change parameters →  → ...

Setting

Data arrives **sequentially** as a stream

$$(x_1, y_1), \ldots, (x_{t-1}, y_{t-1}), (x_t, ?) \in \mathcal{X} \times \mathcal{Y} \subseteq [0, 1] \times \mathbb{R}$$

and we want to predict each next response y_t as follows:

Setting

Data arrives **sequentially** as a stream

$$(x_1, y_1), \ldots, (x_{t-1}, y_{t-1}), (x_t, ?) \in \mathcal{X} \times \mathcal{Y} \subseteq [0,1] \times \mathbb{R}$$

and we want to predict each next response y_t as follows:

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

- observes input $x_t \in \mathcal{X}$
- makes prediction $\hat{y}_t \in \mathcal{Y}$
- incurs loss $\ell_t(y_t, \hat{y}_t)$ with true target $y_t \in \mathcal{Y}$
- updates predictions $\hat{y}_t \to \hat{y}_{t+1}$

Setting

Data arrives **sequentially** as a stream

$$(x_1, y_1), \ldots, (x_{t-1}, y_{t-1}), (x_t, ?) \in \mathcal{X} \times \mathcal{Y} \subseteq [0,1] \times \mathbb{R}$$

and we want to predict each next response y_t as follows:

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

- observes input $x_t \in \mathcal{X}$
- makes prediction $\hat{y}_t \in \mathcal{Y}$
- incurs loss $\ell_t(y_t, \hat{y}_t)$ with true target $y_t \in \mathcal{Y}$
- updates predictions $\hat{y}_t \to \hat{y}_{t+1}$

Choose $\hat{y}_t$ before observing ℓ_t

No assumptions on how ℓ_t is generated!

Classical regression setting:

- $y_t = g(x_t) + W_t$ for some $g : \mathbb{R} \to \mathbb{R}$ and $W_t \sim \mathcal{N}(0, \sigma^2)$
- square loss $\ell_t(y_t, \hat{y}_t) = (y_t - \hat{y}_t)^2$.

Setting

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

- observes input $x_t \in \mathcal{X}$
- makes prediction $\hat{y}_t \in \mathcal{Y}$
- incurs loss $\ell_t(y_t, \hat{y}_t)$ with true target $y_t \in \mathcal{Y}$
- updates predictions $\hat{y}_t \to \hat{y}_{t+1}$

Goal:

minimize the cumulative loss

$$\min_{\hat{y}_1, \ldots, \hat{y}_T} \sum_{t=1}^{T} \ell_t(y_t, \hat{y}_t)$$

Setting

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

- observes input $x_t \in \mathcal{X}$
- makes prediction $\hat{y}_t \in \mathcal{Y}$
- incurs loss $\ell_t(y_t, \hat{y}_t)$ with true target $y_t \in \mathcal{Y}$
- updates predictions $\hat{y}_t \to \hat{y}_{t+1}$

Goal:

minimize the cumulative loss $\Leftrightarrow$ predict almost as well as the best strategy $y^\star$

$$\min_{\hat{y}_1, \dots, \hat{y}_T} \sum_{t=1}^{T} \ell_t(y_t, \hat{y}_t)$$

$$\min_{\hat{y}_1, \dots, \hat{y}_T} \underbrace{\sum_{t=1}^{T} \ell_t\left(y_t, \hat{y}_t\right) - \inf_{y^\star} \sum_{t=1}^{T} \ell_t\left(y_t, y^\star\right)}_{:=\mathrm{Regret}_T(y^\star)}$$

Regret in *Non Parametric* Regression

***Non-Parametric* regression** means that we are interested in forecasters $(\hat{y}_t)$ whose regret

$$\mathrm{Regret}_T(\mathcal{F}) = \underbrace{\sum_{t=1}^{T} \ell_t\left(y_t, \hat{y}_t\right)}_{\text{our performance}} \quad - \quad \underbrace{\inf_{f \in \mathcal{F}} \sum_{t=1}^{T} \ell_t\left(y_t, f(x_t)\right)}_{\text{reference performance}}$$

over some benchmark function class $\mathcal{F} \in \mathcal{Y}^{\mathcal{X}}$ is as **small** as possible.

Regret in *Non Parametric* Regression

***Non-Parametric* regression** means that we are interested in forecasters $(\hat{y}_t)$ whose regret

$$\mathrm{Regret}_T(\mathcal{F}) = \underbrace{\sum_{t=1}^{T} \ell_t\left(y_t, \hat{y}_t\right)}_{\text{our performance}} \quad - \quad \underbrace{\inf_{f \in \mathcal{F}} \sum_{t=1}^{T} \ell_t\left(y_t, f(x_t)\right)}_{\text{reference performance}} \quad \underbrace{= o(T)}_{\text{goal}}$$

over some benchmark function class $\mathcal{F} \in \mathcal{Y}^{\mathcal{X}}$ is as **small** as possible.

Regret in *Non Parametric* Regression

***Non-Parametric* regression** means that we are interested in forecasters $(\hat{y}_t)$ whose regret

$$\mathrm{Regret}_T(\mathcal{F}) = \underbrace{\sum_{t=1}^{T} \ell_t\left(y_t, \hat{y}_t\right)}_{\text{our performance}} - \underbrace{\inf_{f\in\mathcal{F}} \sum_{t=1}^{T} \ell_t\left(y_t, f(x_t)\right)}_{\text{reference performance}} \underbrace{= o(T)}_{\text{goal}}$$

over some benchmark function class $\mathcal{F} \in \mathcal{Y}^{\mathcal{X}}$ is as **small** as possible.

💡 **Solution:** producing prediction as a function of x_t

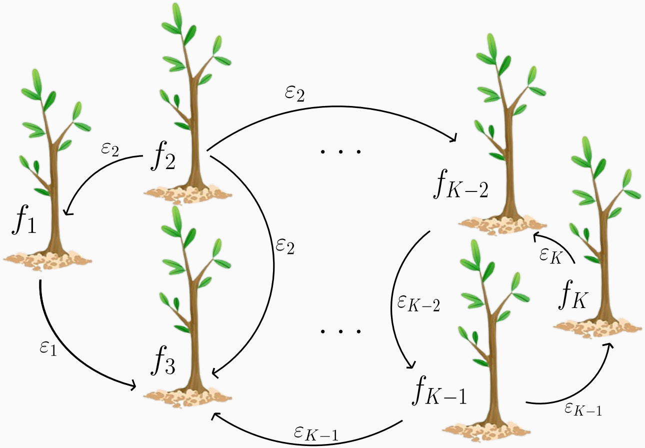
$$\hat{y}_t = F_t(x_t), \quad F_t \in \mathcal{Y}^{\mathcal{X}} \text{ sequentially updated.}$$

Building predictions with Boosting

Boosting uses "wisdom of the crowd"

- **Boosting:** ensemble method combining multiple weak learners to create a strong learner
- Each weak model corrects/learns from errors of its peers

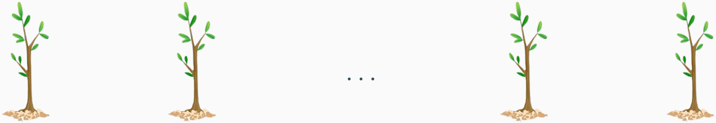
→ Resulting in a **highly accurate** predictive model [1]



[1] e.g. AdaBoost and XGBoost

How to deal with weak learners?

For each $t = 1, \ldots, T$, we use $K \geqslant 1$ *sequential* and *weak* predictors



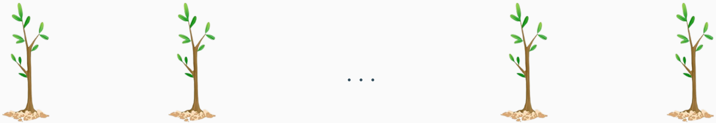
$f_{1,t}$ $f_{2,t}$ $\ldots$ $f_{K-1,t}$ $f_{K,t}$

from a class of *weak learners*

$$\mathcal{W} := \{x \mapsto f(x; \theta, I) : \theta \text{ parameter of } f \text{ with support } I\} \subset \mathcal{Y}^{\mathcal{X}}.$$

How to deal with weak learners?

For each $t = 1, \ldots, T$, we use $K \geqslant 1$ *sequential* and *weak* predictors



$f_{1,t}$ $f_{2,t}$ $\cdots$ $f_{K-1,t}$ $f_{K,t}$

from a class of *weak learners*

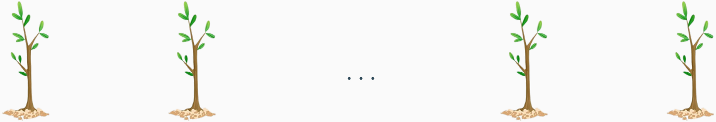
$$\mathcal{W} := \{x \mapsto f(x; \theta, I) : \theta \text{ parameter of } f \text{ with support } I\} \subset \mathcal{Y}^{\mathcal{X}}.$$

Example: $\mathcal{W}_1$ set of regression trees with (low) depth 1,

$\mathcal{W}_1 = \Big\{$  $\Big\} = \{f(\cdot; \theta, I) : \theta \in \mathbb{R}^2 \text{ and } I = (I^{(1)}, I^{(2)}), I^{(1)} \sqcup I^{(2)} = \mathcal{X}\}.$

How to deal with weak learners?

For each $t = 1, \ldots, T$, we use $K \geqslant 1$ *sequential* and *weak* predictors



$f_{1,t}$ $\quad$ $f_{2,t}$ $\quad \ldots \quad$ $f_{K-1,t}$ $\quad$ $f_{K,t}$

We make our predictions at any time $t \geqslant 1$ as

$$\hat{y}_t = F_{K,t}(x_t) = \sum_{k=1}^{K} f_{k,t}(x_t),$$

using the *strong estimator* $F_{K,t} \in \left\{ F_K = \sum_{k=1}^{K} f_k : f_k \in \mathcal{W} \right\} =: \mathrm{span}_K(\mathcal{W})$

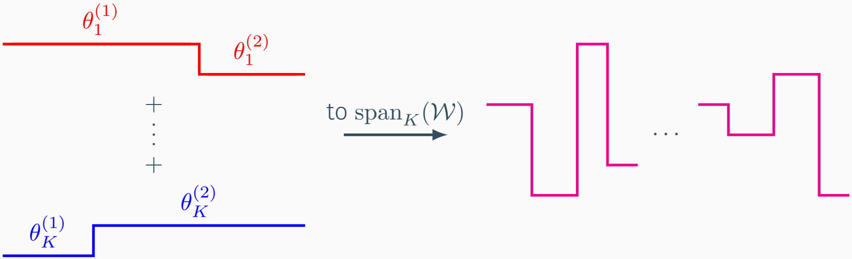
How to deal with weak learners?

We make our predictions at any time $t \geqslant 1$ as

$$\hat{y}_t = F_{K,t}(x_t) = \sum_{k=1}^{K} f_{k,t}(x_t),$$

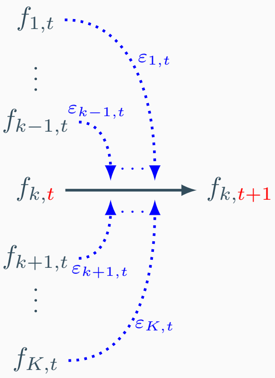
using the *strong estimator* $F_{K,t} \in \left\{ F_K = \sum_{k=1}^{K} f_k : f_k \in \mathcal{W} \right\} =: \mathrm{span}_K(\mathcal{W})$

Example of *strong learner* using weak learners in $\mathcal{W}_1$:



Boosting process

At any time $t \geqslant 1$, for each $k \in [K]$:



❶ every $f_{k,t}$ dicovers x_t and residuals $\varepsilon_{k,t}$,

❷ $f_{k,t}$ receives residuals $\varepsilon_{1,t}, \ldots, \varepsilon_{K,t}$ from others $\{f_{1,t}, \ldots, f_{K,t}\} \backslash \{f_{k,t}\}$ and observes its gradient $g_{k,t} = \nabla_{f_k} \ell_t \left(y_t, \sum_{k=1}^{K} f_{k,t} \right)$,

❸ $f_{k,t}$ is updated in $f_{k,t+1}$ using $g_{k,t}$.

e.g. if $\ell_t(y_t, \hat{y}_t) = (y_t - \hat{y}_t)^2$, residuals are $\varepsilon_{k,t} = y_t - \sum_{l \neq k} f_{l,t}(x_t)$ and gradients are $g_{k,t} = \frac{\partial}{\partial f_k} \ell_t(y_t, \sum_k f_{k,t}) = 2 f'_{k,t}(x_t)(\hat{y}_t - y_t)$

Architecture of our Online Boosting Algorithm

Algorithm 1: Online Boosting

Init: K sequential weak-learners

for $t = 1$ **to** T **do**

  Receive data x_t;

  Predict $\hat{y}_t = F_{K,t}(x_t) = \sum_{k=1}^{K} f_{k,t}(x_t)$;

  Incur $\ell_t(\hat{y}_t, y_t)$, reveal residuals $\varepsilon_{k,t}$ and gradients $g_{k,t} = \nabla_{f_{k,t}} \ell_t(y_t, \sum_k f_{k,t})$ for all $k = 1, \ldots, K$;

  for $k = 1$ **to** K **do**

$$f_{k,t+1} \leftarrow \texttt{update}(f_{k,t}, g_{k,t}) \tag{1}$$

Return: $F_{K,T+1} = \sum_{k=1}^{K} f_{K,T+1}$

Regret Analysis of an Online Boosting Algorithm

Back on Regret Analysis

Assumption: losses (ℓ_t) are convex and differentiable in $\hat{y}_t$

$\rightarrow$ Goal is to optimize in each predictor f_k, so we can rewrite the problem with $\ell_t : \mathcal{W}^K \rightarrow \mathbb{R}$ and

$$\mathrm{Regret}_T(\mathcal{F}) = \sum_{t=1}^{T} \ell_t(f_{1,t}, \dots, f_{K,t}) - \min_{f \in \mathcal{F}} \sum_{t=1}^{T} \ell_t(f)$$

? How to bound above regret?

Back on Regret Analysis

$$\text{Regret}_T(\mathcal{F}) = \sum_{t=1}^{T} \ell_t(f_{1,t}, \dots, f_{K,t}) - \min_{f \in \mathcal{F}} \sum_{t=1}^{T} \ell_t(f)$$

💡 Decompose as a sum of 2 stage regrets:

$$\text{Regret}_T(\mathcal{F}) = \underbrace{\sum_{t=1}^{T} \ell_t(f_{1,t}, \dots, f_{K,t}) - \min_{f_1^\star, \dots, f_K^\star \in \mathcal{W}} \sum_{t=1}^{T} \ell_t(f_1^\star, \dots, f_K^\star)}_{\text{Regret}_T^{(1)} = \text{regret of the algo against the best combination in } \mathcal{W}}$$

$$+ \underbrace{\min_{f_1^\star, \dots, f_K^\star \in \mathcal{W}} \sum_{t=1}^{T} \ell_t(f_1^\star, \dots, f_K^\star) - \min_{f \in \mathcal{F}} \sum_{t=1}^{T} \ell_t(f)}_{\text{Regret}_T^{(2)} = \text{regret of best combination in } \mathcal{W} \text{ against } \mathcal{F}}$$

A first analysis: Regret with OGD

Assume $\{f_1, \ldots, f_K\} = \{\{\theta_1, I_1\}, \ldots, \{\theta_K, I_K\}\}$ are constants on restricted domains $(I_k) \subset \mathcal{X}$.

- *Online Gradient Descent*: online version of Gradient Descent
- Can be applied to any *convex* and *differentiable* loss function
- `update`$(\theta_{k,t}, g_{k,t})$ is

$$\theta_{k,t+1} \leftarrow \Pi_{\Theta_k}(\theta_{k,t} - \eta_{k,t} g_{k,t})$$

  for some sets $(\Theta_k) \subset \mathbb{R}$

A first analysis: Regret with OGD

Assume $\{f_1, \ldots, f_K\} = \{\{\theta_1, I_1\}, \ldots, \{\theta_K, I_K\}\}$ are constants on restricted domains $(I_k) \subset \mathcal{X}$.

- `update`$(\theta_{k,t}, g_{k,t})$ is

$$\theta_{k,t+1} \leftarrow \Pi_{\Theta_k}(\theta_{k,t} - \eta_{k,t} g_{k,t})$$

Theorem (Estimation regret with OGD)

*Assume (ℓ_t) are differentiable for any $k \in [K]$ and for any $t \geqslant 1, \nabla_k \ell_t(\theta_{1,t}, \ldots, \theta_{K,t}) \leqslant G$. Algorithm 1 with **OGD** has regret*

$$\mathrm{Regret}_T^{(1)}(\theta_1^\star, \ldots, \theta_K^\star) \lesssim G \sum_{k=1}^{K} D_k \sqrt{T_k}$$

with $D_k = \sup_{\theta_1, \theta_2 \in \Theta_k} |\theta_1 - \theta_2|$ and $T_k = |\{t : x_t \in I_k\}|$.

A first analysis: Regret with OGD

- `update`$(\theta_{k,t}, g_{k,t})$ is

$$\theta_{k,t+1} \leftarrow \Pi_{\Theta_k}(\theta_{k,t} - \eta_{k,t} g_{k,t})$$

Theorem (Estimation regret with OGD)

*Assume (ℓ_t) are differentiable for any $k \in [K]$ and for any $t \geqslant 1, \nabla_k \ell_t(\theta_{1,t}, \ldots, \theta_{K,t}) \leqslant G$. Algorithm 1 with **OGD** has regret*

$$\mathrm{Regret}_T^{(1)}(\theta_1^\star, \ldots, \theta_K^\star) \lesssim G \sum_{k=1}^{K} D_k \sqrt{T_k}$$

with $D_k = \sup_{\theta_1, \theta_2 \in \Theta_k} |\theta_1 - \theta_2|$ and $T_k = |\{t : x_t \in I_k\}|$.

☹ Θ_k sets? Their size D_k? Tuning $\eta_{k,t}$?
Does not depend optimally to competitors in $\mathcal{W}$

ParameterFree Regret

💡 Consider a *Parameter Free subroutine* [2] in `update`$(\theta_{k,t}, g_{k,t})$

Theorem (Estimation regret with ParamFree)

Assume (ℓ_t) *are differentiable for any* $k \in [K]$ *and for any* $t \geqslant 1, \nabla_k \ell_t(\theta_1, \ldots, \theta_K) \leqslant G$. *Algorithm 1 with* `ParameterFree` *achieves*

$$\mathrm{Regret}_T^{(1)}(\theta_1^\star, \ldots, \theta_K^\star) \lesssim G \sum_{k=1}^{K} |\theta_k^\star| \sqrt{T_k}$$

with $T_k = |\{t : x_t \in I_k\}|$.

[2] Orabona and Pál, "Coin betting and parameter-free online learning".

ParameterFree Regret

Consider a *Parameter Free subroutine* in `update`$(\theta_{k,t}, g_{k,t})$

Theorem (Estimation regret with ParamFree)

Assume (ℓ_t) *are differentiable for any* $k \in [K]$ *and for any* $t \geqslant 1, \nabla_k \ell_t(\theta_1, \ldots, \theta_K) \leqslant G$. *Algorithm 1 with* `ParameterFree` *achieves*

$$\mathrm{Regret}_T^{(1)}(\theta_1^\star, \ldots, \theta_K^\star) \lesssim G \sum_{k=1}^{K} |\theta_k^\star| \sqrt{T_k}$$

with $T_k = |\{t : x_t \in I_k\}|$.

No sets Θ_k! No more learning rate $\eta_{k,t}$ to tune!

Adaptive to optimal size $|\theta_k^\star|$ and works for any weak learners!

Where do we stand?

- We managed to bound *estimation* regret using a `ParameterFree` subroutine
- We obtained regret $\mathcal{O}\left(G \sum_{k=1}^{K} |\theta_k^\star| \sqrt{T_k}\right)$ that *does not depend* on the type of weak models
- This ensures a **diameter adaptive** procedure

💡 We may benefit from empirical decreasing $|\theta_1^\star| \geqslant |\theta_2^\star| \geqslant \ldots \geqslant |\theta_K^\star|$
$\rightarrow F_k$ is becoming more accurate as k grows

- We have

$$\mathrm{Regret}_T(\mathcal{F}) \quad \lesssim \quad G \sum_{k=1}^{K} |\theta_k^\star| \sqrt{T_k} \quad + \quad \underbrace{\mathrm{Regret}_T^{(2)}}_{\text{approximation regret}}$$

The case of Lipschitz functions

Let us take $\mathcal{F}$ the set of $L-$lipschitz function on $\mathcal{X} = [0,1]$ i.e. for $f \in \mathcal{F}$,

$$\forall x_1, x_2 \in \mathcal{X}, \quad |f(x_1) - f(x_2)| \leqslant L|x_1 - x_2|.$$

$\rightarrow$ We want to best approximate any competitor $f \in \mathcal{F}$ with $F_K \in \mathrm{span}_K(\mathcal{W})$.

⚠ Approximation regret *depends* on the type of weak learners, e.g. if $\mathrm{span}_K(\mathcal{W}) \approx \mathcal{F}$ hence small approx. regret

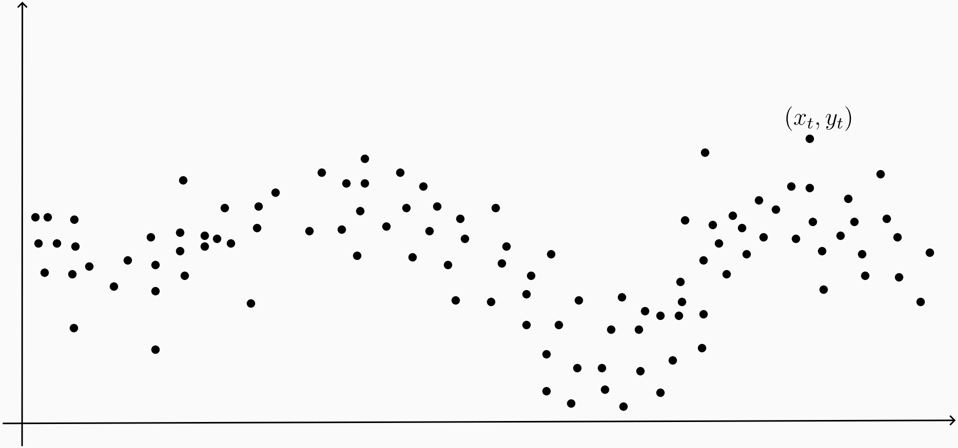
Boosting with Dyadic Trees in $\mathcal{W}_1$

- Assume the following process: launch a dyadic regression tree from $\mathcal{W}_1$ in each node until depth is $M \geqslant 1$

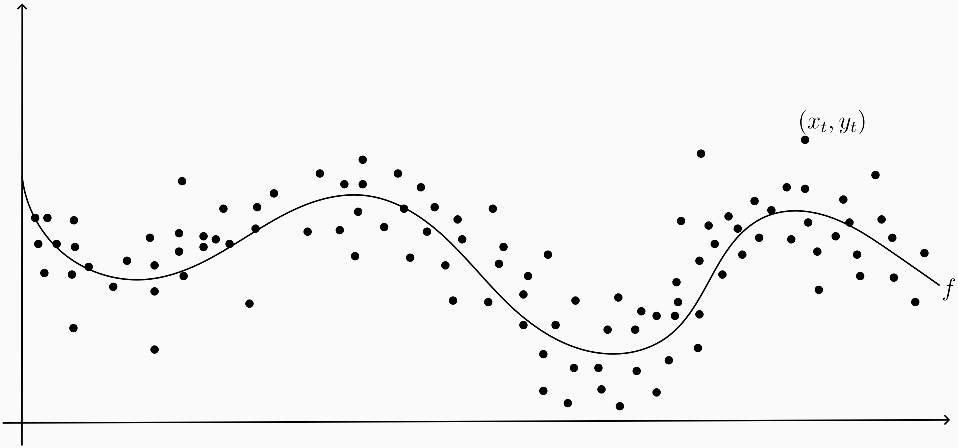
- Dyadic scheme $\Rightarrow$ we have $|\theta_k^\star| \leqslant \frac{L}{2^{m_k}}$ with $m_k = m$ if $k \in [\![2^{m-1}, 2^m - 1]\!]$

- For ℓ_t square loss, we have the following illustration:

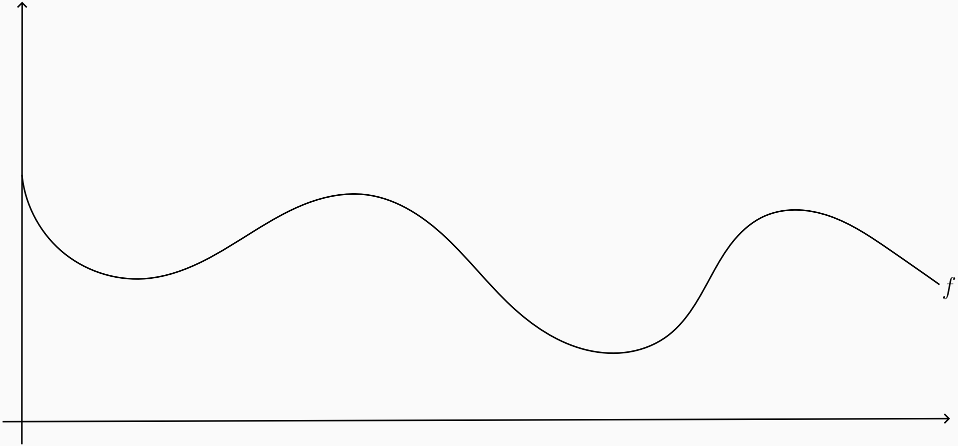
Boosting with Dyadic Trees in $\mathcal{W}_1$



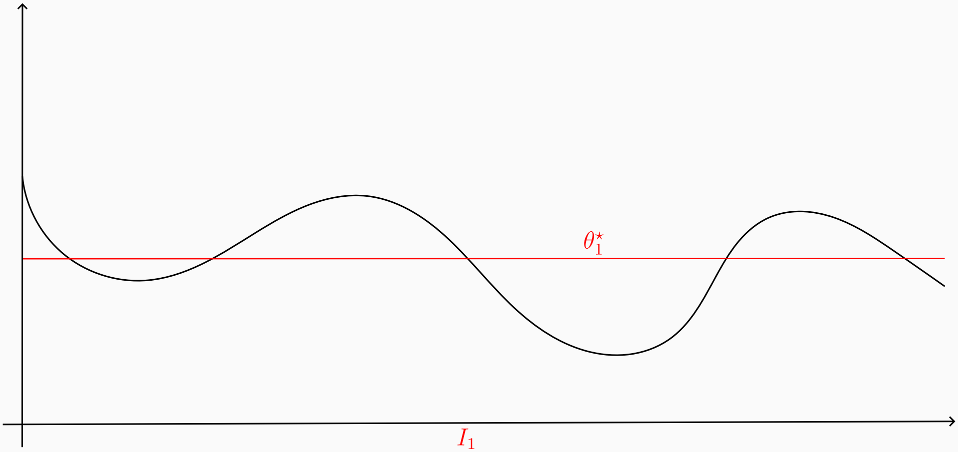
Boosting with Dyadic Trees in $\mathcal{W}_1$



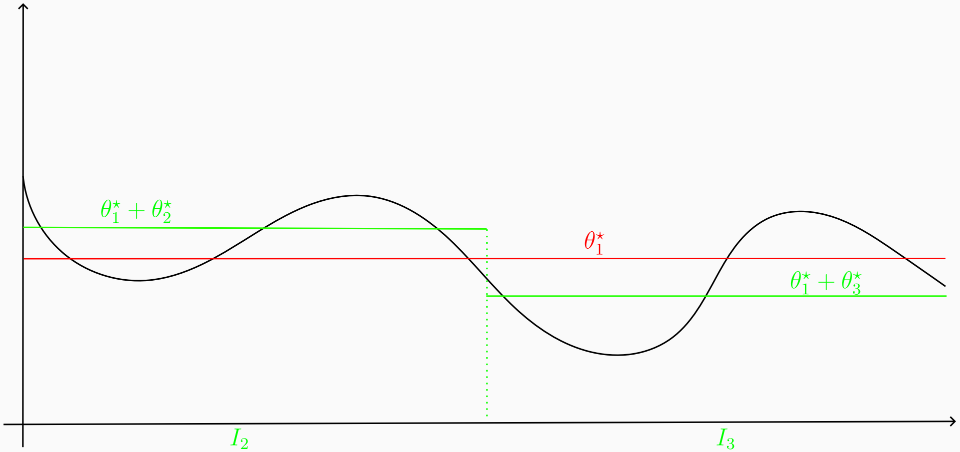
Boosting with Dyadic Trees in $\mathcal{W}_1$



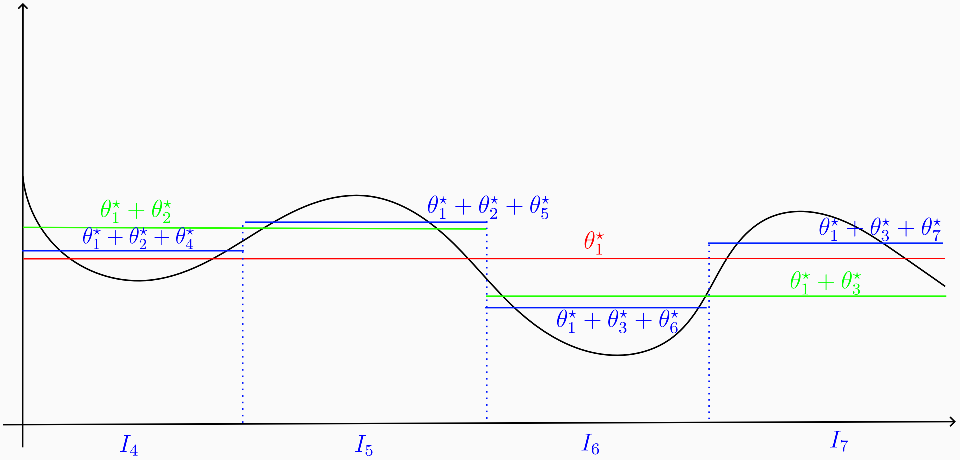
Boosting with Dyadic Trees in $\mathcal{W}_1$



Boosting with Dyadic Trees in $\mathcal{W}_1$



Boosting with Dyadic Trees in $\mathcal{W}_1$



Final Regret

Theorem

Let $\mathcal{F}$ be the set of L-Lipschitz function, $M \approx \log_2(T)$ and ℓ_t be the square or absolute loss. Our Algorithm 1 with Dyadic Trees in $\mathcal{W}_1$ has regret

$$\mathrm{Regret}_T(\mathcal{F}) \lesssim GL\sqrt{T}$$

🍀 Computationally tractable: x_t only falls into one subinterval I_k for each level $m \in [M]$: we update $\mathcal{O}(T \log_2(T))$ for T rounds.

Final Regret

Theorem

Let $\mathcal{F}$ be the set of L-Lipschitz function, $M \approx \log_2(T)$ and ℓ_t be the square or absolute loss. Our Algorithm 1 with Dyadic Trees in $\mathcal{W}_1$ has regret

$$\mathrm{Regret}_T(\mathcal{F}) \lesssim GL\sqrt{T}$$

🍀 Computationally tractable: x_t only falls into one subinterval I_k for each level $m \in [M]$: we update $\mathcal{O}(T \log_2(T))$ for T rounds.

? Can we do better?

Perspectives

- Although sublinear: we want $\mathrm{Regret}_T(\mathcal{F}) = \mathcal{O}(T^{1/2}) \longrightarrow \mathcal{O}(T^{1/3})$ for square loss (minimax)
- Designing Locally-Lipschitz adaptive algorithm with Boosting

Thank you!

Questions?

References

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

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.

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.

Experiments

Simulations

Consider the following model:

$$y_t = \cos(3\pi x) - \sin(3x) + W_t, \quad W_t \sim \mathcal{N}(0, 0.5)$$