How to Boost Any Loss Function

Richard Nock

Yishay Mansour

Google Research

Summary

- Two popular ML optimization frameworks have taken opposite trajectories:
  - (S)**G**D started by optimization using **G**radients and recently moved to 0^{th} order optimization with just loss function queries
  - Boosting started as a "native" 0^{th} framework (no gradient usage assumed) but a substantial % of field quickly geared towards **G**radient boosting
- Little is known on what loss functions can be optimized in boosting's original framework, *i.e.* using a barely-better-than-random oracle, a *weak learner*
- Important question not just for boosting: all convergence rates for (S)**G**D ⟶ 0^{th} order make assumptions about loss itself (cvx, diff., Lip., smooth, etc.)

Google Research

Summary

- Two popul ajectories:
  - (S)**G**D moved to 0th order
  - Boost sage assumed) but a boosting
- Little is kn ng's original framework
- Important order make assumptio

↪ Our paper settles the question: any loss with discontinuities forming a set of 0 Lebesgue measure (computer-wise, it means any loss)

↪ Our proof is constructive: we give an algorithm

↪ Boosting (convergence) rate has the optimal $1/\gamma^2$ dependence in the weak learner's advantage over random guessing γ

“Quantum calculus”, Kac & Cheung 2002.

Key tool

- At the core, (S)**G**D ⟶ 0th order replaces gradient with secant slope

$$\delta_v F(z) \doteq \frac{F(z+v) - F(z)}{v}$$

offset

- This = *h*-derivative in *quantum calculus* (calculus without derivatives), a field that also uses higher order quantities with several times the same offset
- Need a more general 1+-order notion where offsets can be a (multi)set:

$$\delta_{\mathcal{V}} F(z) \doteq \begin{cases} F(z) & \text{if} \quad \mathcal{V} = \emptyset \\ \delta_v F(z) & \text{if} \quad \mathcal{V} = \{v\} \\ \delta_{\{v\}}(\delta_{\mathcal{V}\backslash\{v\}} F)(z) & \text{otherwise} \end{cases}$$

Google Research

Nock & Mansour, “How to Boost Any Loss Function”, NeurIPS’24

“Quantum calculus”, Kac & Cheung 2002.

Key tool

- At the
- This … a field that a … set
- Need a more general 1⁺-order notion where offsets can be a (multi)set:

$$\delta_{\mathcal{V}} F(z) \doteq \begin{cases} F(z) & \text{if} \quad \mathcal{V} = \emptyset \\ \delta_v F(z) & \text{if} \quad \mathcal{V} = \{v\} \\ \delta_{\{v\}}(\delta_{\mathcal{V}\backslash\{v\}} F)(z) & \text{otherwise} \end{cases}$$

Example, with two offsets, generalizes 2nd order derivative

$$\delta_{\{b,c\}} F(a) = \frac{2}{b} \cdot \frac{1}{c} \cdot \left(\frac{F(a+b+c) + F(a)}{2} - \frac{F(a+b) + F(a+c)}{2} \right)$$

(if F convex, then $\delta_{\{b,c\}} F(a) \geqslant 0$)

Google Research

Nock & Mansour, “How to Boost Any Loss Function”, NeurIPS’24

Boosting: key facts

- Architecture à-la-AdaBoost:
  - Linear combination, $H_T = \sum_{t \in [T]} \alpha_t h_t$
  - Each dimension $\leftarrow$ weak classifier
  - Leveraging coefficients (α_t) computed during boosting
- Differences / generalization:
  - Weighting scheme for example and sample fed to weak learner
  - Each offset $\leftarrow$ *new oracle*

Google Research

Key parts of the algorithm / generalization wrt boosting

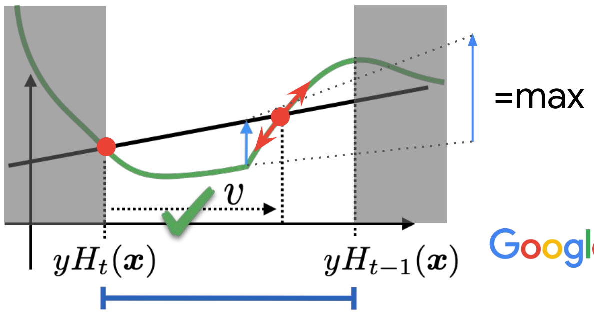
- Weight vector at iteration *t*+1 of the form $\boldsymbol{w}_{t+1} = -[\delta_{v_{ti}} F(y_i H_t(\boldsymbol{x}_i))]_i$

  ↪ weights can be negative (all-positive iff *F* non-increasing)
- Sample for weak learner at iteration *t* is $\mathcal{S}_t \doteq \{(\boldsymbol{x}_i, y_i \cdot \mathrm{sign}(w_{ti}))\}_i$ (and weights $|\boldsymbol{w}_t|$)

  ↪ labels can be flipped
- Need an **offset oracle** that provides at each iteration *t* the set of offsets $\{v_{ti}\}_i$

  ↪ any *v* such that the max elevation (secant - *F*) in interval defined by last edges does not exceed a *specific bound*

  (in gradient boosting, *v*=0)



Google Research

Leveraging coefficients – general case

- The "*specific bound*" for offsets **and** the leveraging coefficient require a > 0 upperbound $\overline{w}_{2,t}$ on a 2nd order *v*-derivative (curvature-like) parameter, i.e.:

$$\mathbb{E}_{i\sim[m]}\left[\delta_{\{\alpha_t y_i h_t(\boldsymbol{x}_i), v_{(t-1)i}\}} F(y_i H_{t-1}(\boldsymbol{x}_i)) \cdot \left(\frac{h_t(\boldsymbol{x}_i)}{M_t}\right)^2\right] \leqslant \overline{w}_{2,t}$$

Tricky bit: contains the leveraging coefficient !

$M_t \doteq \max_i |h_t(\boldsymbol{x}_i)|$

Google Research

Nock & Mansour, *"How to Boost Any Loss Function", NeurIPS'24*

Leveraging coefficients – **easy** case

- Can be easy to get a "nice" value if *F* has special properties
  - e.g. *F* β-smooth ⇒ can pick $\overline{w}_{2,t} = 2\beta$
  - in such cases, *range* of boosting-compliant leveraging coefficients:

$$\alpha_t \in \frac{\eta_t}{2(1+\varepsilon_t)M_t^2\overline{w}_{2,t}} \cdot [1-\pi_t, 1+\pi_t]$$

important for boosting rate

  - η_t = expected empirical edge, M_t = max absolute weak learning prediction
  - ε_t, π_t user-fixed such that $\varepsilon_t > 0, \pi_t \in (0,1)$ (the smaller, the better)

Google Research

Nock & Mansour, "How to Boost Any Loss Function", NeurIPS'24

Leveraging coefficients – **hard** case

- Otherwise, efficient algorithm giving all parameters at once ($\alpha_t, \varepsilon_t, \overline{w}_{2,t}$ & π_t)

(Our boosting algorithm is called SecBoost, see paper for details)

Boosting !

- Let the expected empirical loss of classifier *H* be $F(\mathcal{S}, H) \doteq \mathsf{E}_{i\sim[m]}[F(y_i H(\boldsymbol{x}_i))]$ and its initial value (first constant classifier, e.g. 0) $F_0 \doteq F(\mathcal{S}, h_0)$.
- Then, for any $z \in \mathbb{R}$ s.t. $F(z) \leqslant F_0$, if SecBoost is run for #*T* iterations sat.

$$T \geqslant \frac{4(F_0 - F(z))}{\gamma^2 \rho} \cdot \frac{1 + \max_t \varepsilon_t}{1 - \max_t \pi_t^2}$$

then $F(\mathcal{S}, H_T) \leqslant F(z)$, assuming the following assumptions:

ϱ-Weak Convergence Regime

$$\frac{|\mathbb{E}_{i\sim[m]}[w_{ti}]|^2}{\overline{w}_{2,t}} \geqslant \rho > 0$$

weights carry "information"

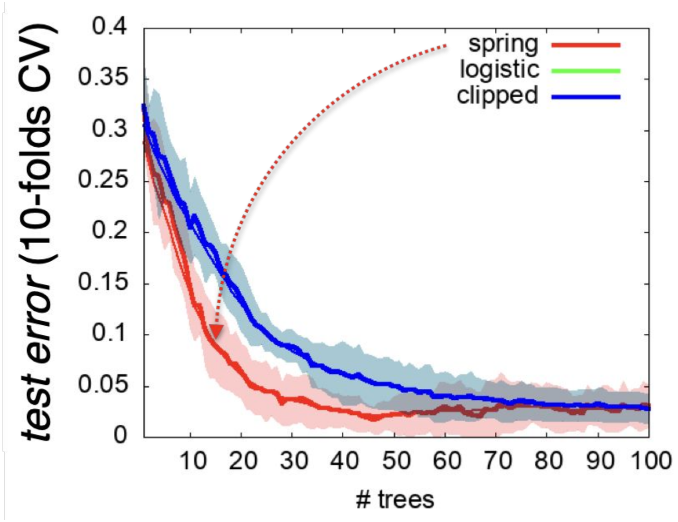
loss "jiggling" (→local mins.)

γ-Weak Learning Assumption

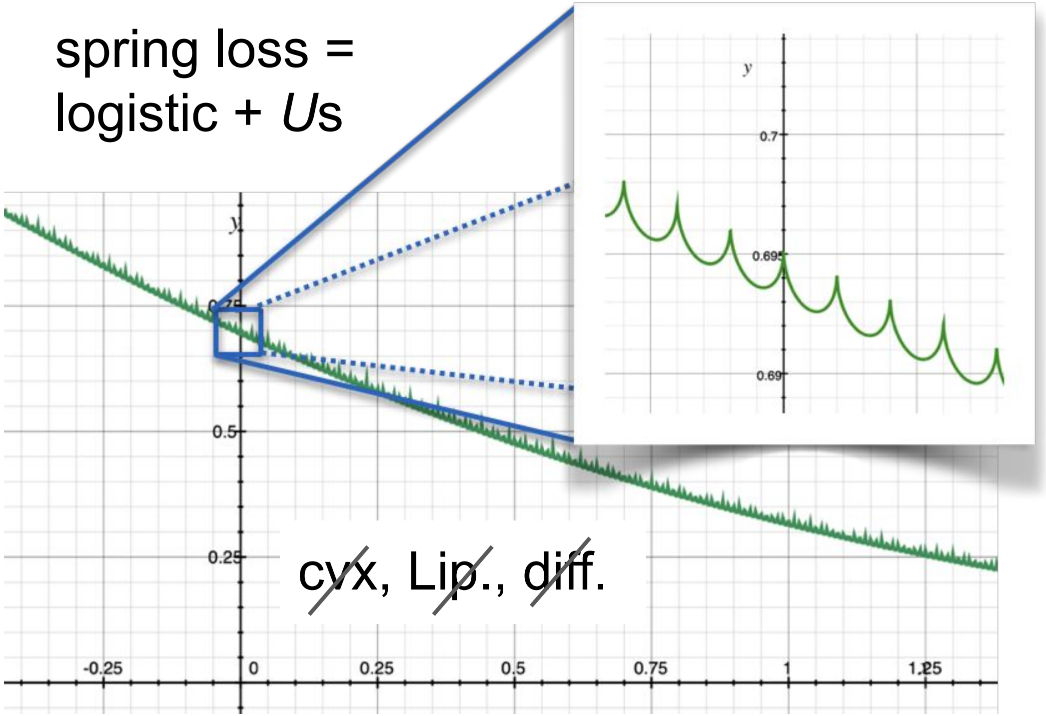
$$\left|\mathbb{E}_{\tilde{\boldsymbol{w}}_t}\left[\tilde{y}_{ti} \cdot \frac{h_t(\boldsymbol{x}_i)}{M_t}\right]\right| \geqslant \gamma > 0$$

Google Research

Toy Experiment



spring loss =
logistic + *U*s



~~cvx~~, ~~Lip.~~, ~~diff.~~

Google Research

Nock & Mansour, "How to Boost Any Loss Function", NeurIPS'24

Thank You

Richard Nock — Google Research
Yishay Mansour — Tel Aviv University and Google Research