OpReg-Boost: Learning to Accelerate Online Algorithms with Operator Regression

Anonymous Authors[1]

Abstract

In this paper we propose a learning-based acceleration scheme for online optimization. .... novel operator regression ...

1. Introduction

.... problem formulation, literature review, contribution statement .....

2. Operator Regression

2.1. Problem formulation

Consider the following *convex, time-varying* optimization problem

$$\boldsymbol{x}^*(t) \in \arg\min f(\boldsymbol{x}; t) + g(\boldsymbol{x}; t) \tag{1}$$

where $f : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}$ and $g : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R} \cup \{+\infty\}$ are closed, convex and proper functions, optionally with $g \equiv 0$. In particular, we are interested in solving the following sequence of static problems derived from sampling (1) at the times $\{t_k\}_{k\in\mathbb{N}}$, $t_{k+1} - t_k = T_\mathrm{s}$:

$$\boldsymbol{x}^*(t) \in \arg\min f(\boldsymbol{x}; t_k) + g(\boldsymbol{x}; t_k). \tag{2}$$

Problem (2) is convex but not strongly so, which means that the performance of online algorithms applied to it is worse than the performance attainable for strongly convex problems. As an example, consider the case $g \equiv 0$, and apply an online gradient descent: if the problem is convex the (static) regret is $O(\sqrt{T})$, with T the number of sampling times, while if the problem is strongly convex the regret is $O(\log(T))$, section 3.1 in (Hazan, 2016). Instead of citing Hazan, we should derive a result in an appendix, for example for the fixed point residual

As we can see from the results just mentioned, in general in online optimization it is not possible to achieve zero regret, due to the dynamic nature of the problem. However, for strongly convex problems smaller regrets can be achieved.

The goal then is to design learning techniques that allow to achieve strongly convex-like performance while tracking a solution trajectory of the problem (2) with good accuracy.

2.2. Operator regression

First of all, we define the observations

$$\boldsymbol{y}_i = \mathcal{T}_f \boldsymbol{x}_i + \boldsymbol{e}_i, \quad i \in [D]$$

of the operator we want to regularize. Now, using Fact 2.2 in (Ryu et al., 2020) we know that an operator $\mathcal{T}$ is ζ-contractive interpolable if and only if it satisfies

$$\|\mathcal{T}\boldsymbol{x}_i - \mathcal{T}\boldsymbol{x}_j\|^2 \leq \zeta^2 \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2, \quad \forall i, j \in [D],\ i \neq j.$$

Therefore, we can define the following constrained regression problem:

$$\begin{aligned} \hat{\boldsymbol{t}}_i = &\arg\min_{\boldsymbol{t}_i \in \mathbb{R}^n} \frac{1}{2} \sum_{i\in[D]} \|\boldsymbol{t}_i - \boldsymbol{y}_i\|^2 \\ &\text{s.t. } \|\boldsymbol{t}_i - \boldsymbol{t}_j\|^2 \leq \zeta^2 \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \ \forall i, j \in [D], i \neq j \end{aligned} \tag{3}$$

where again the cost function is a least square terms on the observations and the constraints enforce contractiveness.

Auto-tuning contraction We can modify (3) by including $w = \zeta^2$ as an unknown of the regression problem, which becomes

$$\begin{aligned} \hat{\boldsymbol{t}}_i = &\arg\min_{\boldsymbol{t}_i \in \mathbb{R}^n} \frac{1}{2} \sum_{i\in[D]} \|\boldsymbol{t}_i - \boldsymbol{y}_i\|^2 + \frac{c}{2} w^2 \\ &\text{s.t. } \|\boldsymbol{t}_i - \boldsymbol{t}_j\|^2 - w \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \leq 0 \ \forall i, j \in [D], i \neq j \end{aligned} \tag{4}$$

this way the contraction constant does not need to be specified, and is *auto-tuned* by the regression.

PRS-based solver .... brief description of basic characteristics, reference to the appendix ....

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

3. OpReg-Boost

Let now $\mathcal{T}_k : \mathbb{R}^n \to \mathbb{R}^n$ be a solver for the convex problem observed at time t_k, and assume that the operator has a term $\mathcal{T}_{f,k}$ depending exclusively on f (*e.g.* we have $\mathcal{T}_k = \mathcal{T}'_k \circ \mathcal{T}_{f,k}$). Since the sampled problem is convex, the solver $\mathcal{T}_k$ is non-expansive (actually, averaged, to guarantee convergence).

The idea then is to try and learn an approximation of $\mathcal{T}_k$ that is *contractive* rather than non-expansive.

Again, w.l.o.g. we are only interested in learning the term $\mathcal{T}_{f,k}$ that depends on f. The algorithm can be described as follows: at each time t_k do:

1. sample a new problem, that is, observe $f(\boldsymbol{x}; t_k)$ and $g(\boldsymbol{x}; t_k)$;
2. approximate the term $\mathcal{T}_{f,k}$ of the solver with a contractive one, and
3. apply the resulting solver. For example if $\mathcal{T}_k = \mathcal{T}'_k \circ \mathcal{T}_{f,k}$, then we apply $\mathcal{T}_k = \mathcal{T}'_k \circ \hat{\mathcal{T}}_{f,k}$ where $\hat{\mathcal{T}}_{f,k}$ is the learned operator.

The learning step is performed using the (novel) constrained operator regression OpReg described in the following.

In particular we have the following algorithm make this pseudo code For $\ell \in 0, 1, \dots, M$ (for some $M \in \mathbb{N}$):

- let $\boldsymbol{x}_k^\ell$ be the current approximate solution, we choose $D-1$ points around it, *e.g.*

$$\boldsymbol{x}_i = \boldsymbol{x}_k^\ell + \boldsymbol{p}_i,\ i = 2, \dots, D,\ \boldsymbol{p}_i \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}_n)$$

  and set $\boldsymbol{x}_1 = \boldsymbol{x}_k^\ell$;
- we compute the data $\boldsymbol{t}_i = \mathcal{T}_{f,k}\boldsymbol{x}_i$, $i = 1, \dots, D$, for example $\boldsymbol{t}_i = \boldsymbol{x}_i - \alpha \nabla f(\boldsymbol{x}_i; t_k)$;
- we solve the OpReg, and use the approximate operator at $\boldsymbol{x}_k^\ell$, that is $\boldsymbol{t}_1$, in the chosen solver $\mathcal{T}_k$:

$$\boldsymbol{x}_k^{\ell+1} = \operatorname{prox}_{\alpha g}(\boldsymbol{t}_1)$$

  where α is a step-size.

3.1. Interpolated version

4. Numerical Results

4.1. Simulations set-up

We consider the following time-varying problem:

$$\boldsymbol{x}^*(t_k) = \operatorname*{arg\,min}_{\boldsymbol{x} \in \mathbb{R}^n} \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}(t_k)\|^2 + w \|\boldsymbol{x}\|_1 \tag{5}$$

with $n = 10$, $\boldsymbol{A}$ matrix with maximum and minimum (non-zero) eigenvalues $\sqrt{L} = 10^8$, $\sqrt{\mu} = 1$, and with rank 5; $\boldsymbol{y}(t_k)$ has sinusoidal components with 3 zero components. Due to $\boldsymbol{A}$ being rank deficient, the cost f is convex but not strongly so.

4.2. Results

References

Hazan, E. Introduction to Online Convex Optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

Ryu, E. K., Taylor, A. B., Bergeling, C., and Giselsson, P. Operator Splitting Performance Estimation: Tight Contraction Factors and Optimal Parameter Selection. *SIAM Journal on Optimization*, 30(3):2251–2271, 2020.

Simonetto, A. Smooth Strongly Convex Regression. In *2020 28th European Signal Processing Conference (EUSIPCO)*, pp. 2130–2134, Amsterdam, January 2021. IEEE.

A. PRS-Based QCQP solver

In this section we present a solver for OpReg which can be efficiently parallelized, inspired by the approach in (Simonetto, 2021).

The idea is as follows: each pair of data points $i, j \in [D]$, $i \neq j$, gives rise to one constraint, for a total of $D(D-1)/2$ constraints. We define the following set of pairs

$$\mathcal{V} = \{e = (i, j) \mid i, j \in [D],\ i < j\}$$

which are ordered (that is, for example we take $(1, 2)$ and not $(2, 1)$, to avoid counting constraints twice). Clearly to each pair $e = (i, j)$ corresponds the constraint $\|\boldsymbol{t}_i - \boldsymbol{t}_j\|^2 \leq \zeta^2 \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$.

Let now $\boldsymbol{t}_{i,e}$ and $\boldsymbol{t}_{j,e}$ be copies of $\boldsymbol{t}_i$ and $\boldsymbol{t}_j$ associated to the e-th constraint; then we can equivalently reformulate the OpReg (3) as

$$\min_{\boldsymbol{t}_{i,e}, \boldsymbol{t}_{j,e}} \frac{1}{2(D-1)} \sum_{e \in \mathcal{V}} \left\| \begin{bmatrix} \boldsymbol{t}_{i,e} \\ \boldsymbol{t}_{j,e} \end{bmatrix} - \begin{bmatrix} \boldsymbol{y}_i \\ \boldsymbol{y}_j \end{bmatrix} \right\|^2 \tag{6a}$$

$$\text{s.t. } \|\boldsymbol{t}_{i,e} - \boldsymbol{t}_{j,e}\|^2 \leq \zeta^2 \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \tag{6b}$$

$$\boldsymbol{t}_{i,e} = \boldsymbol{t}_{i,e'}\ \forall e, e' | i \sim e, e'. \tag{6c}$$

Clearly (6) is a strongly convex problem with convex constraints defined in the variables $\boldsymbol{t}_{i,e}$.

A.1. PRS solver

Let $\boldsymbol{\xi}$ be the vector stacking all the $\boldsymbol{t}_{i,e}$, then the problem is equivalent to

$$\min_{\boldsymbol{\xi}} f(\boldsymbol{\xi}) + g(\boldsymbol{\xi})$$

where

$$f(\boldsymbol{\xi}) = \frac{1}{2(D-1)} \|\boldsymbol{\xi} - \boldsymbol{y}\|^2 + f_1(\boldsymbol{\xi})$$

with f_1 the indicator function imposing (6b) and g the indicator function imposing the "consensus" constraints (6c). The problem can then be solved using the Peaceman-Rachford splitting (PRS) characterized by the following updates $\ell \in \mathbb{N}$:

$$\boldsymbol{\xi}^\ell = \operatorname{prox}_{\rho f}(\boldsymbol{z}^\ell) \tag{7a}$$

$$\boldsymbol{v}^\ell = \operatorname{prox}_{\rho g}(2\boldsymbol{\xi}^\ell - \boldsymbol{z}^\ell) \tag{7b}$$

$$\boldsymbol{z}^{\ell+1} = \boldsymbol{z}^\ell + \boldsymbol{v}^\ell - \boldsymbol{\xi}^\ell. \tag{7c}$$

The proximal of g corresponds to the projection onto the consensus space, and thus can be characterized simply by

$$\boldsymbol{v}_{i,e}^\ell = \frac{1}{D-1} \sum_{e' | i \sim e'} \left(2\boldsymbol{t}_{i,e'}^\ell - \boldsymbol{z}_{e'}^\ell\right).$$

Regarding the proximal of f, is is clear that f is separable, in the sense that it can be written as

$$f(\boldsymbol{\xi}) = \sum_{e \in \mathcal{V}} \left[\frac{1}{2(D-1)} \left\| \begin{bmatrix} \boldsymbol{t}_{i,e} \\ \boldsymbol{t}_{j,e} \end{bmatrix} - \begin{bmatrix} \boldsymbol{y}_i \\ \boldsymbol{y}_j \end{bmatrix} \right\|^2 + \iota_e(\boldsymbol{t}_{i,e}, \boldsymbol{t}_{j,e}) \right]$$

where ι_e denotes the indicator function of (6b). Therefore, the update (7a) can be solved by solving (possibly in parallel) the problems

$$(\boldsymbol{t}_{i,e}, \boldsymbol{t}_{j,e}) = \arg\min \left\{ \frac{1}{2(D-1)} \left\| \begin{bmatrix} \boldsymbol{t}_{i,e} \\ \boldsymbol{t}_{j,e} \end{bmatrix} - \begin{bmatrix} \boldsymbol{y}_i \\ \boldsymbol{y}_j \end{bmatrix} \right\|^2 + \frac{1}{2\rho} \left\| \begin{bmatrix} \boldsymbol{t}_{i,e} \\ \boldsymbol{t}_{j,e} \end{bmatrix} - \boldsymbol{z}_e^\ell \right\| \right\}$$
$$\text{s.t.} \quad \|\boldsymbol{t}_{i,e} - \boldsymbol{t}_{j,e}\|^2 \leq \zeta^2 \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2. \tag{8}$$

A.2. Local updates

The problems (8) are quadratic programs with quadratic constraints, that is, they can be written in the form

$$\min_{\boldsymbol{\xi}} \frac{1}{2}\boldsymbol{\xi}^\top \boldsymbol{P}_0 \boldsymbol{\xi} + \langle \boldsymbol{q}_0, \boldsymbol{\xi} \rangle \tag{9a}$$

$$\text{s.t. } \frac{1}{2}\boldsymbol{\xi}^\top \boldsymbol{P}_1 \boldsymbol{\xi} + \langle \boldsymbol{q}_1, \boldsymbol{\xi} \rangle + r_1 \leq 0. \tag{9b}$$

In particular, for the cost function we have

$$\boldsymbol{P}_0 = \left(\frac{1}{D-1} + \frac{1}{\rho}\right) \boldsymbol{I}_{2n}, \quad \boldsymbol{q}_0 = -\left(\frac{1}{D-1} \begin{bmatrix} \boldsymbol{y}_i \\ \boldsymbol{y}_j \end{bmatrix} + \frac{1}{\rho} \boldsymbol{z}_e^\ell\right)$$

and for the constraint

$$\boldsymbol{P}_1 = 2 \begin{bmatrix} \boldsymbol{I}_n & -\boldsymbol{I}_n \\ -\boldsymbol{I}_n & \boldsymbol{I}_n \end{bmatrix}, \quad \boldsymbol{q}_1 = 02n, \quad r_1 = -\zeta^2 \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2.$$

B. Interpolation Using MAP