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

Anonymous Authors[1]

Abstract

We present OpReg-Boost, a novel acceleration scheme to boost the convergence properties and lessen the asymptotical error of online algorithms for time-varying (weakly) convex optimization problems. OpReg-Boost is built to learn the closest algorithm to a given online algorithm that has all the algorithmic properties one needs for fast convergence, and it is based on the concept of operator regression. We show how to build OpReg-Boost by using a Peaceman-Rachford solver, and further boost [... interpolation-MAP..]. Simulation results showcase the [better/increase/..] properties of OpReg-Boost w.r.t. the more classical [...], and its close relative convex-regression-boost which is also novel but significantly less performing.

1. Introduction

[Add others' papers, esp. ML community]

In recent years, we have witnessed the growing interest in time-varying optimization problems, where the cost function, constraints, or both are parametrized over data streams, and therefore are changing over time (Jadbabaie et al., 2015; Dall'Anese et al., 2020). Notable applications for the machine learning community are subspace tracking for video streaming and online sparse subspace clustering, see (Asif & Romberg, 2014; Akhriev et al., 2020; Dall'Anese et al., 2020) and references therein.

Solving time-varying optimization problems with online algorithms amounts to find and track continuously changing optimizers. If we let the optimizers trajectory be defined as $\boldsymbol{x}^*(t)$, a significant results for online algorithms in this context is that if the *path-length* $\sum_{k=1}^{T} \|\boldsymbol{x}^*(t_{k+1}) - \boldsymbol{x}^*(t_k)\|$, for subsequent time instances t_{k+1}, t_k, is not sub-linear in T, then online algorithms will have an asymptotical error $O(1)$ (Besbes et al., 2015; Jadbabaie et al., 2015; Dall'Anese et al., 2020; Li et al., 2020) (or $O(T)$ in terms of pertinent notions of regret)[1]. While errors can be made small, for instance with the use of prediction-correction algorithms (Dontchev et al., 2013; Simonetto et al., 2020), the existence of an asymptotical error remains a distinctive feature of online algorithms for time-varying optimization.

[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.

An eye-opening intuition is then to use the existence of said error to one's advantage, by introducing regularizations in the problem formulation that boost the algorithms convergence. In time-varying scenarios, where convergence rate is crucial, often time then one obtains smaller asymptotical error w.r.t. the original problem if one uses regularizations than if they do not. To reiterate, surprisingly, often when one uses pertinent regularizations, *there is no trade-off between accuracy and speed* and regularized problems offer better speed and asymptotical errors w.r.t. the original problem, even though we are modifying the problem formulation (Simonetto & Leus, 2014; Bastianello et al., 2020).

By building on this field-defining fact, once can ask what is the best regularization for a time-varying problem at hand? In this paper, we explore this question in two slightly different angles. First, we ask ourselves:

(Q1) *what is the closest problem to a given time-varying problem that has all the functional properties we need for fast convergence?*

This gives rise to convex-regression-based boosting.

To fix the ideas in a static setting, let us consider Figure 1, where we have depicted a non-convex function f, which we can evaluate at specific points (grey dot). The idea behind Q1 and convex-regression-based boosting is then to interpret the functional evaluation as noisy measurements of an underlying true convex function $\hat{f}$, which we want to learn. As long as f and $\hat{f}$ are not dramatically different, the reasoning is that solving the problem of minimizing $\hat{f}$ instead of f will then give us a boost in term of convergence rate and asymptotical error in online algorithm, despite the fact that f and $\hat{f}$ are different. In addition, and for

[1]In (Jadbabaie et al., 2015) and subsequent works, [Finish]

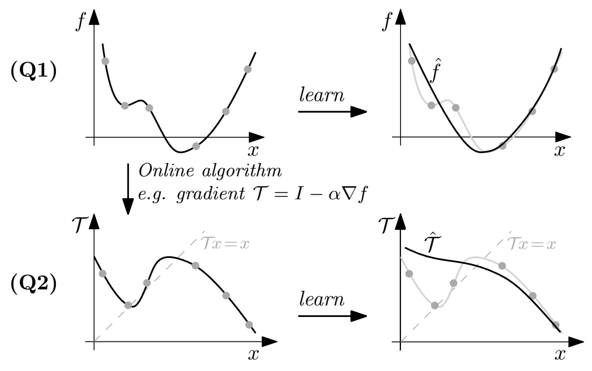

Figure 1. The idea of boosting via projection onto the space of "good" functions or "good" fixed point operators. One can interpret the evaluation of function f or operator $\mathcal{T}$ as noisy evaluations of an underlying "better" function or operator, $\hat{f}$ and $\hat{\mathcal{T}}$, respectively, and use the latter to solve the problem instead. This gives rise to convex-regression-based boosting or operation-regression-based boosting (OpReg-Boost).

free, since we have some liberty in designing the functional properties of $\hat{f}$ (e.g., Lipschitz constant, strong convexity constant, etc.), we can thoroughly optimize the parameter of the online algorithm, e.g., its stepsize, further boosting convergence.

Convex regression is however not the only way to learn good problems, and not the best one (as we are going to explore in the simulation section). A better way is operator regression, which stems from the second question we ask:

(Q2) *what is the closest algorithm to a given online algorithm for solving a time-varying problem that has all the algorithmic properties we need for fast convergence?*

In Figure 1, we have reported the case of a gradient descent algorithm in terms of a fixed point operator $\mathcal{T} = I - \alpha\nabla_x f$, with $\alpha > 0$ being the stepsize. The idea here is to use evaluations of $\mathcal{T}$ as noisy measurements of a true underlying operator $\hat{\mathcal{T}}$, with useful properties (e.g., a contractive operator). By using $\hat{\mathcal{T}}$ en lieu of $\mathcal{T}$, then one will be able to boost convergence and reduce the asymptotical error, once again, if $\mathcal{T}$ and $\hat{\mathcal{T}}$ are not dramatically different.

This latter methodology offers the most promise (as we will showcase in simulation) and it is what we call OpReg-Boost (boosting by operator regression).

In this paper, we offer the following contributions,

1. We present two novel regression methods to *learn-project-and-solve* time-varying optimization problems that are not necessarily convex. The methods are based on convex regression and operator regression, and – especially the latter – is promising in boosting convergence without introducing a larger asymptotical error.
2. We present efficient ways to solve the operation regression problem via a pertinent reformulation of the Peaceman-Rachford splitting method. [.. more]
3. We [Interpolation]
4. We showcase the [simulation results]

1.1. Literature review

Time-varying optimization and online algorithms are extensively covered in (Dall'Anese et al., 2020; Simonetto et al., 2020) and references therein, while a more machine learning approach is offered in (Jadbabaie et al., 2015) and subsequent work. The notion that regularizations help convergence possibly without introducing extra asymptotical error is presented in the papers (Simonetto & Leus, 2014; Bastianello et al., 2020). While we refer to (Devolder et al., 2012; Koshal et al., 2011) for seminal papers close in spirit in the static domain.

Learning the optimize and regularize is a growing research topic, and we cite (Nghiem et al., 2018; Ongie et al., 2020), as related work, even though not on the problem we are interested in solving here.

Convex regression is treated extensively in (Seijo & Sen, 2011; Lim & Glynn, 2012; Mazumder et al., 2019; Blanchet et al., 2019), while recently being generalized to smooth strongly convex functions (Simonetto, 2021) based on A. Taylor's works (Taylor et al., 2016; Taylor, 2017).

Operator regression is a recent and at the same time old topic. We are going to build on the recent work (Ryu et al., 2020) and the F.A. Valentine's 1945 paper (Valentine, 1945).

The acceleration schemes that we compare with are [Tell and cite] e.g., AA: (Mai & Johansson, 2020a)

1.2. Notation

Notation is wherever possible standard. We use the concept of μ-weakly convex function, to indicate a function $f : \mathbb{R}^n \to \mathbb{R}$ that becomes convex by adding the term $\frac{\mu}{2}\|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2$, $\mu > 0$ (see (Duchi & Ruan, 2018; Davis & Drusvyatskiy, 2019; Mai & Johansson, 2020b)). The set of convex functions on $\mathbb{R}^n$ that are L-smooth (i.e., have L-Lipschitz continuous gradient) and m-strongly convex is indicated with $\mathcal{S}_{m,L}(\mathbb{R}^n)$.

An operator $\mathcal{T} : \mathbb{R}^n \to \mathbb{R}^n$ is said to be non-expansive iff $\|T\boldsymbol{x} - T\boldsymbol{y}\| \leq \|\boldsymbol{x} - \boldsymbol{y}\|$, for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, whereas it is ζ-contractive, $\zeta \in (0,1)$, iff $\|T\boldsymbol{x} - T\boldsymbol{y}\| \leq \zeta\|\boldsymbol{x} - \boldsymbol{y}\|$, for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$.

The prox operator [..]

2. Problem Formulation

We are interested in solving the following time-varying optimization problem,

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

where $f : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}$ is closed, proper, and μ-weakly convex, whereas $g : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R} \cup \{+\infty\}$ is closed, convex and proper function, optionally with $g \equiv 0$. Upon sampling the problem at discrete time instances t_k, we then obtain the sequence of time-invariant problems,

$$\boldsymbol{x}^*(t_k) \in \arg\min_{\boldsymbol{x}} f(\boldsymbol{x};t_k) + g(\boldsymbol{x};t_k), \qquad k \in \mathbb{N}. \tag{2}$$

Our overarching goal is to set up an online algorithm $\mathcal{A}$ that generate a sequence of approximate optimizers $\{\boldsymbol{x}_k\}_{k\in\mathbb{N}}$, such that the asymptotical tracking error

$$\limsup_{k\to\infty} \|\boldsymbol{x}_k - \boldsymbol{x}^*(t_k)\|, \tag{3}$$

is as small as possible. Our main blanket assumption is that optimizers[2] at subsequent time instances cannot be arbitrarily far apart, that is,

$$\|\boldsymbol{x}^*(t_{k+1}) - \boldsymbol{x}^*(t_k)\| \leq \mathit{\Delta} < +\infty, \qquad k \in \mathbb{N}, \tag{4}$$

which in turn guarantees that the path-length grows linearly in time,

$$\sum_{k\in\mathbb{N}} \|\boldsymbol{x}^*(t_{k+1}) - \boldsymbol{x}^*(t_k)\| \leq \mathit{\Delta} k. \tag{5}$$

The first important and known result here is that if function f is in $\mathcal{S}_{m,L}(\mathbb{R}^n)$ uniformly in k, then an online algorithm A (here a forward-backward algorithm) can obtain linear convergence to the asymptotical error bound $\mathit{\Delta}/(1-\gamma)$, with optimal $\gamma = \frac{L-m}{L+m}$; much weaker results hold for the general convex case (Simonetto, 2017), for example [..]

The second important result is that even a simple Tikhonov regularization $+\frac{w}{2}\|x\|_2^2$, transforming a smooth but not strongly convex case into a strongly convex one allows for a boost in convergence and [..]

With this in place, our two main research questions are

- **Q1.** Can we project the weakly convex function f onto $\mathcal{S}_{m,L}(\mathbb{R}^n)$ and learn a better function $\hat{f}$ that has all the functional properties we need (e.g., smooth strong convexity) and use that to solve the problem instead?
- **Q2.** Can we project the fixed point operator of the algorithm we use $\mathcal{T}$ onto the set of contracting operators, learn a better operator $\hat{\mathcal{T}}$, and use that to solve the problem instead?

[2] ..

3. Q1. Convex Regression

We briefly introduce here the concept of convex regression, while we leave the main technical details to (Mazumder et al., 2019; Simonetto, 2021). Suppose one has collected ℓ noisy measurements of a convex function $\varphi(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ (say y_i) at points $\boldsymbol{x}_i \in \mathbb{R}^n$, $i \in I_\ell$. Then convex regression is a least-squares approach to estimate the generating function based on the measurements. Formally, letting $\varphi \in \mathcal{S}_{m,L}(\mathbb{R}^n)$, then one would like to solve the infinite-dimensional problem,

$$\hat{\varphi}_\ell \in \arg\min_{\psi\in\mathcal{S}_{m,L}(\mathbb{R}^n)} \left\{ \sum_{i\in I_\ell} (y_i - \psi(\boldsymbol{x}_i))^2 \right\}. \tag{6}$$

The problem can be then equivalently decomposed into an estimation on the data points, to find the true function values and gradients at the data point ($\boldsymbol{f} = [\varphi_i]_{i\in I_\ell}$, $\boldsymbol{\delta} = [\nabla\varphi_i]_{i\in I_\ell}$) which turns out to be a convex quadratically constrained quadratic program,

$$(\boldsymbol{f}^*, \boldsymbol{\delta}^*) \quad = \arg\min_{\boldsymbol{f}\in\mathbb{R}^\ell,\ \boldsymbol{\delta}\in\mathbb{R}^{n\ell}} \sum_{i\in I_\ell} (y_i - \varphi_i)^2 \tag{7a}$$

$$\begin{aligned} \text{s.t.}: \quad & \varphi_i - \varphi_j - \boldsymbol{\delta}_j^\top(\boldsymbol{x}_i - \boldsymbol{x}_j) \geq \\ & \frac{1}{2(1-m/L)}\left(\frac{1}{L}\|\boldsymbol{\delta}_i - \boldsymbol{\delta}_j\|_2^2 + m\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 \right. \\ & \left. -2\frac{m}{L}(\boldsymbol{\delta}_j - \boldsymbol{\delta}_i)^\top(\boldsymbol{x}_j - \boldsymbol{x}_i)\right), \quad \forall i,j \in I_\ell. \end{aligned} \tag{7b}$$

And an interpolation scheme, that extends the point estimate over the whole space maintaining the functional properties,

$$\hat{\varphi}_\ell(\boldsymbol{x}) = \mathrm{conv}(p_i(\boldsymbol{x})) + \frac{m}{2}\|\boldsymbol{x}\|_2^2 \in \mathcal{S}_{m,L}(\mathbb{R}^n), \tag{8}$$

where

$$\begin{aligned} p_i(\boldsymbol{x}) := \frac{L-m}{2}\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2 + (\boldsymbol{\delta}_i^* - m\boldsymbol{x}_i)^\top\boldsymbol{x} + \\ - \boldsymbol{\delta}_i^{*,\top}\boldsymbol{x}_i + f_i^* + m/2\|\boldsymbol{x}_i\|_2^2, \end{aligned} \tag{9}$$

and where $\mathrm{conv}(\cdot)$ indicates the convex hull.

Then, to solve Q1, one could consider a number of evaluations of the function $f(\boldsymbol{x};t_k)$ as noisy measurements of an underlying convex function $\varphi_k \in \mathcal{S}_{m,L}(\mathbb{R}^n)$ and using the least-squares approach above to *project* function $f(\cdot;t_k)$ onto the space of "good" functions [Complete.. put the details in the appendix]

4. Q2. Operator Regression

A slightly different approach to convex regression is to look at the algorithm directly, instead of the function. First of all, we remind the reader that any algorithm can be seen as an operator that maps the current approximate optimizer

$\boldsymbol{x}_k$ into a new approximate optimizer $\boldsymbol{x}_{k+1}$. For example, a gradient algorithm of the form,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k; t_k) \equiv \underbrace{(I - \alpha \nabla_x f_k)}_{=:\mathcal{T}_k}(\boldsymbol{x}_k) \quad (10)$$

where $\mathcal{T}_k : \mathbb{R}^n \to \mathbb{R}^n$ is the gradient algorithm operator. Interpreting algorithms as operators (possibly averaged, monotone, etc.) has been extremely fruitful for characterizing their convergence properties (Rockafellar, 1976; Eckstein, 1989; Bauschke & Combettes, 2017; Ryu & Boyd, 2016; **?**).

The counterpart of a smooth strongly convex function φ_k in convex regression, is an operator that is contracting, i.e., $\|\mathcal{T}_k \boldsymbol{x} - \mathcal{T}_k \boldsymbol{y}\| \leq \zeta \|\boldsymbol{x} - \boldsymbol{y}\|$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and $\zeta \in (0, 1)$. In fact, while L-smooth m-strongly convex functions give rise to gradient operators that are ..-contracting, the space of contracting operators is larger (i.e., a function f that is not strongly convex could in principle give rise to a contracting operator (**?**)).

The key idea now is to interpret evaluations of $\mathcal{T}_k$ as measurements of a true underlying contracting operator that we want to estimate.

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

$$\|\mathcal{T}\boldsymbol{x}_i - \mathcal{T}\boldsymbol{x}_j\|^2 \leq \zeta^2 \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2, \quad \forall i, j \in I_\ell,\ i \neq j. \quad (11)$$

Therefore, we can define the following convex quadratically constrained quadratic program as our regression problem:

$$\begin{aligned} \hat{\boldsymbol{t}} = \underset{\mathbb{R}^{n\ell} \ni \boldsymbol{t} = [\boldsymbol{t}_i]_{i \in I_\ell}}{\arg\min} \ & \frac{1}{2} \sum_{i \in I_\ell} \|\boldsymbol{t}_i - \mathcal{T}_k \boldsymbol{x}_i\|^2 \\ \text{s.t.} \ & \|\boldsymbol{t}_i - \boldsymbol{t}_j\|^2 \leq \zeta^2 \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \ \forall i, j \in I_\ell, i \neq j, \end{aligned} \quad (12)$$

where again the cost function is a least square terms on the "observations"' and the constraints enforce contractiveness. In particular, the optimal values $\hat{\boldsymbol{t}}$ on the data points represent the evaluations of a ζ-contracting operator when applied to those points.

To extend [interpolation here]

4.1. Optional auto-tuning

Discretionarily, we can also modify (12) by including $w = \zeta^2$ as an unknown of the regression problem with a penalty $c > 0$, which then becomes

$$\begin{aligned} (\hat{\boldsymbol{t}}, \hat{w}) = \underset{\mathbb{R}^{n\ell} \ni \boldsymbol{t} = [\boldsymbol{t}_i]_{i \in I_\ell}, w \in (0,1)}{\arg\min} \ & \frac{1}{2} \sum_{i \in I_\ell} \|\boldsymbol{t}_i - \boldsymbol{y}_i\|^2 + \frac{c}{2} w^2 \\ \text{s.t.} \ & \|\boldsymbol{t}_i - \boldsymbol{t}_j\|^2 - w \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \leq 0 \ \forall i, j \in I_\ell, i \neq j, \end{aligned} \quad (13)$$

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

4.2. PRS-based solver

[Add here]

5. OpReg-Boost

We are now ready to present our main algorithm.

Having to deal with composite problems $f(\cdot; t_k) + g(\cdot; t_k)$, we focus here on online algorithms of the forward-backward type, i.e.,

$$\boldsymbol{x}_{k+1} = \text{prox}_{g_k}(\boldsymbol{x}_k - \alpha \nabla_{\boldsymbol{x}} f_k(\boldsymbol{x}_k)), \quad k \in \mathbb{N}, \alpha > 0. \quad (14)$$

We let $\mathcal{T}_k = I - \alpha \nabla_{\boldsymbol{x}} f_k$ be the operator we want to regularize. Since f is not smooth strongly convex in general, $\mathcal{T}_k$ is not contractive (in general). However, since g_k is convex, the prox operator is non-expansive, so that if $\mathcal{T}_k$ were to be contractive then the forward-backward algorithm would be contracting itself and we could have better convergence guarantees.

Therefore, we are interested in learning a contracting $\hat{\mathcal{T}}_k$ by evaluations of $\mathcal{T}_k$. The OpReg-Boost 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. Learn the closest contracting operator to $\mathcal{T}_k$, say $\hat{\mathcal{T}}_k$ and
3. Apply $\boldsymbol{x}_{k+1} = \text{prox}_{g_k}(\hat{\mathcal{T}}_k \boldsymbol{x}_k)$.

The learning step is performed using the (novel) constrained operator regression described in (**??**), specifically:

- Choose ℓ points around $\boldsymbol{x}_k$ including $\boldsymbol{x}_k$ itself (e.g., adding a zero-mean Gaussian noise term)
- Evaluate the operator on the data points: $\boldsymbol{t}_i = \mathcal{T}_k \boldsymbol{x}_i$, $i \in I_\ell$, for example $\boldsymbol{t}_i = \boldsymbol{x}_i - \alpha \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_i; t_k)$;
- Solve (12) with the PRS-based solver, and output $\hat{\boldsymbol{t}}_k = \hat{\mathcal{T}}_k \boldsymbol{x}_k$ to be used in Step 3. of OpReg-Boost.

[Some closing words, no need for interpolation?]

5.1. Interpolated version

[Add here]

6. Numerical Results

Wish-list

- non-negative Least-squares

- L1 ?
- Relative-entropy non-negative regression
- weakly convex: time-varying phase retrival? (see Convergence of a Stochastic Gradient Method with Momentum for Nonsmooth Nonconvex Optimization)

Compare (real-time, gradient/functional calls) with Nesterov, AA, CvxReg-Boost, OpReg-Boost, Gradient descent. Other? (Forward-backward envelope, superMann?)

For the weakly convex one: See Convergence of a Stochastic Gradient Method with Momentum for Nonsmooth Nonconvex Optimization (momentum as described there?)

6.1. Simulations set-up

We consider the following time-varying problem:

$$\boldsymbol{x}^*(t_k) = \arg\min_{\boldsymbol{x} \in \mathbb{R}^n} \frac{1}{2} \left\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}(t_k) \right\|^2 + w \left\| \boldsymbol{x} \right\|_1 \tag{15}$$

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.

6.2. Results

References

Akhriev, A., Marecek, J., and Simonetto, A. Pursuit of Low-Rank Models of Time-Varying Matrices Robust to Sparse and Measurement Noise. In *Proceedings of AAAI*, 2020.

Asif, M. S. and Romberg, J. Sparse recovery of streaming signals using ℓ_1-homotopy . *IEEE Transactions on Signal Processing*, 62(16):4209 – 4223, 2014.

Bastianello, N., Simonetto, A., and Carli, R. Distributed Prediction-Correction ADMM for Time-Varying Convex Optimization. In *Proceedings of the Asilomar Conference on Signals, Systems, and Computers*, 2020.

Bauschke, H. H. and Combettes, P. L. *Convex analysis and monotone operator theory in Hilbert spaces*. CMS books in mathematics. Springer, Cham, 2 edition, 2017.

Besbes, O., Gur, Y., and Zeevi, A. Non-stationary Stochastic Optimization. *Operations research*, 63(5):1227 – 1244, 2015.

Blanchet, J., Glynn, P. W., Yan, J., and Zhou, Z. Multivariate Distributionally Robust Convex Regression under Absolute Error Loss. In *Proceedings of NeurIPS*, 2019.

Dall'Anese, E., Simonetto, A., Becker, S., and Madden, L. Optimization and Learning with Information Streams: Time-varying Algorithms and Applications. *Signal Processing Magazine*, May 2020.

Davis, D. and Drusvyatskiy, D. Stochastic Model-Based Minimization of Weakly Convex Functions. *SIAM Journal on Optimization*, 29(1):207–239, 2019.

Devolder, O., Glineur, F., and Nesterov, Y. Double Smoothing Technique for Large-Scale Linearly Constrained Convex Optimization. *SIAM Journal on Optimization*, 22(2): 702 – 727, 2012.

Dontchev, A. L., Krastanov, M. I., Rockafellar, R. T., and Veliov, V. M. An Euler-Newton Continuation method for Tracking Solution Trajectories of Parametric Variational Inequalities. *SIAM Journal of Control and Optimization*, 51(51):1823 – 1840, 2013.

Duchi, J. and Ruan, F. Stochastic Methods for Composite and Weakly Convex Optimization Problems. *SIAM Journal on Optimization*, 28(4):3229–3259, 2018.

Eckstein, J. *Splitting Methods for Monotone Operators with Applications to Parallel Optimization*. PhD thesis, MIT, June 1989.

Jadbabaie, A., Rakhlin, A., Shahrampour, S., and Sridharan, K. Online Optimization: Competing with Dynamic

Comparators. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, PMLR*, number 38, pp. 398 – 406, 2015.

Koshal, J., Nedić, A., and Shanbhag, U. Y. Multiuser Optimization: Distributed Algorithms and Error Analysis. *SIAM Journal on Optimization*, 21(3):1046 – 1081, 2011.

Li, Y., Qu, G., and Li, N. Online Optimization with Predictions and Switching Costs: Fast Algorithms and the Fundamental Limit. *arXiv: 1801.07780*, 2020.

Lim, E. and Glynn, P. W. Consistency of Multidimensional Convex Regression. *Operation Research*, 60(1):196 – 208, 2012.

Mai, V. and Johansson, M. Anderson Acceleration of Proximal Gradient Methods. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 6620–6629, 2020a.

Mai, V. and Johansson, M. Convergence of a Stochastic Gradient Method with Momentum for Non-Smooth Non-Convex Optimization. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 6630–6639, 2020b.

Mazumder, R., Choudhury, A., Iyengar, G., and Sen, B. A Computational Framework for Multivariate Convex Regression and Its Variants. *Journal of the American Statistical Association*, 114(525):318–331, 2019.

Nghiem, T., Stathopoulos, G., and Jones, C. Learning Proximal Operators with Gaussian Processes. In *Proceedings of the Allerton Conference on Communication, Control, and Computing*, 2018.

Ongie, G., Jalal, A., C.A. Metzler, R. B., Dimakis, A., and Willett, R. Deep Learning Techniques for Inverse Problems in Imaging. *IEEE Journal on Selected Areas in Information Theory*, 5, 2020.

Rockafellar, R. T. Monotone Operators and the Proximal Point Algorithm. *SIAM Journal of Control and Optimization*, 14(5):877 – 898, 1976.

Ryu, E. K. and Boyd, S. Primer on Monotone Operator Methods. *Applied Computational Mathematics*, 15(1):3 – 43, 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.

Seijo, E. and Sen, B. Nonparametric Least Squares Estimation of a Multivariate Convex Regression Function. *The Annals of Statistics*, 39(3):1633 – 1657, 2011.

Simonetto, A. Time-Varying Convex Optimization via Time-Varying Averaged Operators . *arXiv: 1704.07338v1*, 2017.

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

Simonetto, A. and Leus, G. Double Smoothing for Time-Varying Distributed Multi-user Optimization. In *Proceedings of the IEEE Global Conference on Signal and Information Processing*, Atlanta, US, December 2014.

Simonetto, A., Dall'Anese, E., Paternain, S., Leus, G., and Giannakis, G. B. Time-Varying Convex Optimization: Time-Structured Algorithms and Applications. *Proceedings of the IEEE*, 108(11):2032 – 2048, 2020.

Taylor, A. *Convex Interpolation and Performance Estimation of First-order Methods for Convex Optimization*. PhD thesis, Université catholique Louvain, Belgium, January 2017.

Taylor, A., Hendrickx, J., and Glineur, F. Smooth Strongly Convex Interpolation and Exact Worst-case Performance of First-order Methods. *Mathematical Programming*, 2016.

Valentine, F. A. A Lipschitz condition preserving extension for a vector function. *American Journal of Mathematics*, 67(1):83–93, 1945.

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 (12) as

$$\min_{\boldsymbol{t}_{i,e}, \boldsymbol{t}_{j,e}} \frac{1}{2(D-1)} \sum_{e \in \mathcal{V}} \left\| \begin{bmatrix} \boldsymbol{t}_{i,e} \\ \boldsymbol{t}_{j,e} \end{bmatrix} - \begin{bmatrix} \boldsymbol{y}_i \\ \boldsymbol{y}_j \end{bmatrix} \right\|^2 \tag{16a}$$

$$\text{s.t. } \|\boldsymbol{t}_{i,e} - \boldsymbol{t}_{j,e}\|^2 \leq \zeta^2 \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \tag{16b}$$

$$\boldsymbol{t}_{i,e} = \boldsymbol{t}_{i,e'}\ \forall e, e' | i \sim e, e'. \tag{16c}$$

Clearly (16) 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 (16b) and g the indicator function imposing the "consensus" constraints (16c). The problem can then be solved using the Peaceman-Rachford splitting (PRS) characterized by the following updates $\ell \in \mathbb{N}$:

$$\boldsymbol{\xi}^\ell = \operatorname{prox}_{\rho f}(\boldsymbol{z}^\ell) \tag{17a}$$

$$\boldsymbol{v}^\ell = \operatorname{prox}_{\rho g}(2\boldsymbol{\xi}^\ell - \boldsymbol{z}^\ell) \tag{17b}$$

$$\boldsymbol{z}^{\ell+1} = \boldsymbol{z}^\ell + \boldsymbol{v}^\ell - \boldsymbol{\xi}^\ell. \tag{17c}$$

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 (16b). Therefore, the update (17a) can be solved by solving (possibly in parallel) the problems

$$(\boldsymbol{t}_{i,e}, \boldsymbol{t}_{j,e}) = \arg\min \left\{ \frac{1}{2(D-1)} \left\| \begin{bmatrix} \boldsymbol{t}_{i,e} \\ \boldsymbol{t}_{j,e} \end{bmatrix} - \begin{bmatrix} \boldsymbol{y}_i \\ \boldsymbol{y}_j \end{bmatrix} \right\|^2 + \frac{1}{2\rho} \left\| \begin{bmatrix} \boldsymbol{t}_{i,e} \\ \boldsymbol{t}_{j,e} \end{bmatrix} - \boldsymbol{z}_e^\ell \right\| \right\}$$
$$\text{s.t.} \quad \|\boldsymbol{t}_{i,e} - \boldsymbol{t}_{j,e}\|^2 \leq \zeta^2 \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2. \tag{18}$$

A.2. Local updates

The problems (18) are quadratic programs with quadratic constraints, that is, they can be written in the form

$$\min_{\boldsymbol{\xi}} \frac{1}{2}\boldsymbol{\xi}^\top \boldsymbol{P}_0 \boldsymbol{\xi} + \langle \boldsymbol{q}_0, \boldsymbol{\xi} \rangle \tag{19a}$$

$$\text{s.t. } \frac{1}{2}\boldsymbol{\xi}^\top \boldsymbol{P}_1 \boldsymbol{\xi} + \langle \boldsymbol{q}_1, \boldsymbol{\xi} \rangle + r_1 \leq 0. \tag{19b}$$

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