Safeguarded Learned Convex Optimization

Anonymous Authors[1]

Abstract

Many applications require repeatedly solving a certain type of optimization problem, each time with new (but similar) data. Data-driven algorithms can "learn to optimize" (L2O) with much fewer iterations and with similar cost per iteration as general-purpose optimization algorithms. L2O algorithms are often derived from general-purpose algorithms, but with the inclusion of (possibly many) tunable parameters. Exceptional performance has been demonstrated when the parameters are optimized for a particular distribution of data. Unfortunately, it is impossible to ensure all L2O algorithms *always* converge to a solution. However, we present a framework that uses L2O updates together with a safeguard to guarantee convergence for convex problems with proximal and/or gradient oracles. The safeguard is simple and computationally cheap to implement, and it should be activated only when the current L2O updates would perform poorly or appear to diverge. This approach yields the numerical benefits of employing machine learning methods to create rapid L2O algorithms while still guaranteeing convergence. Our numerical examples demonstrate the efficacy of this approach for existing and new L2O schemes.

1. Introduction

Solving scientific computing problems often requires application of efficient and scalable optimization algorithms. Data-driven algorithms can execute in much fewer iterations and with similar cost per iteration as state-of-the-art general purpose algorithms. Inspired by one such algorithm called ISTA, (Gregor & LeCun, 2010) proposed treating the entries in fixed matrices/vectors of the algorithm as learnable parameters that can vary by iteration. These entries were then fine-tuned to obtain optimal performance on their data set for a fixed number of iterations. Empirically, their approach converged and showed roughly a 20-fold reduction in computational cost compared to the original algorithm. Several related works followed, also demonstrating numerical success (discussed below). These open the door to a new class of algorithms and analyses. Indeed, classic optimization results often provide worst-case convergence rates, and limited theory exists pertaining to instances drawn from a common distribution (e.g., data supported on a low-dimensional manifold). That is, most L2O methods have little or no convergence guarantees, especially on data distinct from what is seen in training. How then should we balance the desires to use data-driven algorithms and to provide convergence guarantees? We partially address this inquiry by answering the related question:

> *Can a safeguard be added to L2O algorithms to improve robustness and convergence guarantees without significantly hindering performance?*

Here a safeguard is anything that identifies when a "bad" L2O update would occur and what to do in place of that "bad" update. We provide an affirmative answer to the question for convex problems with gradient and/or proximal oracles by proposing such a safeguard and replacing "bad" L2O updates with operations from general-purpose methods.

We establish convergence using any choice among several practical safeguarding procedures. Since we seek a good trade-off between per iteration costs and ensuring convergence, it is essential to be clear about what constitutes a "practical" safeguard in the L2O setting. Three primary factors should be considered: (i) the safeguard must be implementable with known quantities related to all convex problems (e.g., objective values, norms of gradients, and/or the distance between successive iterates); (ii) the L2O and safeguarded L2O schemes should perform identically on "good" data with comparable per-iteration costs; (iii) the safeguard should kick in intermittently and only when "bad" L2O updates would otherwise occur.

The challenge is to create a simple safeguard that kicks in only when needed. Unlike classic optimization algorithms, exceptional L2O algorithms do *not* necessarily exhibit the behavior that each successive iterate is "better" than the current iterate. Loosely speaking, this means there are cases

[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.

where an L2O scheme that gets "worse" for a couple iterates yields a better final output than an L2O scheme that is required to get "better" at each iterate. The intuition behind why this should be acceptable is that one may be solely interested in the final output of the L2O algorithm, not the intermediate steps. From this, we deduce the safeguard should exhibit a form of trailing behavior, i.e., it should provide a measure of progress of previous iterates and only require that, on average, updates are "good." If the safeguard follows too closely, then the safeguarded L2O scheme's flexibility and performance are limited. But, if it follows from too far, then the safeguarded L2O scheme may exhibit highly oscillatory behavior and converge too slowly. The appropriate amount for the safeguard to follow can be estimated by tuning L2O parameters for optimal performance on a training set *without* safeguarding and then using a validation set to test various safeguards with the L2O scheme.

In addition to L2O updates, our method uses a safeguard condition and the update formula from a conventional algorithm. When the safeguard condition holds, the L2O update is used; when it fails, the formula from the conventional algorithm is used. In the ideal case, L2O updates are often used and the conventional algorithm formula provides a "fallback" for exceptional cases. This fallback is designed together with the safeguard condition to ensure convergence. This also implies, even when an L2O algorithm has a fixed number of iterations with tunable parameters, the algorithm may be extended to an arbitrary number of iterations by applying the fallback to compute latter updates.

Review of L2O Methods. A seminal L2O work in the context of sparse coding was by Gregor & LeCun (2010). Numerous follow-up papers also demonstrated empirical success at constructing rapid regressors approximating iterative sparse solvers, compression, ℓ_0 encoding, combining sparse coding with clustering models, nonnegative matrix factorization, compressive sensing MRI, and other applications (Sprechmann et al., 2015; Wang et al., 2016a;b;c;d; Hershey et al., 2014; Yang et al., 2016). A summary of unfolded optimization procedures for sparse recovery is given by Ablin et al. (2019) in Table A.1. The majority of L2O works pertain to sparse coding and provide limited theoretical results. Some works have interpreted LISTA in various ways to provide proofs of different convergence properties (Giryes et al., 2018; Moreau & Bruna, 2017). Others have investigated structures related to LISTA (Xin et al., 2016; Blumensath & Davies, 2009; Borgerding et al., 2017; Metzler et al., 2017), providing results varying by assumptions. (Chen et al., 2018) introduced necessary conditions for the LISTA weight structure to asymptotically achieve a linear convergence rate. This was followed by (Liu et al., 2019a), which proved linear convergence of their ALISTA method for the LASSO problem and provided a result stating that, with high probability, the convergence rate of LISTA is at most linear. The mentioned results are useful, yet can require intricate assumptions and proofs specific to the sparse coding problems.

L2O works have also taken other approaches. For example, the paper by Li & Malik (2016) used reinforcement learning with an objective function f and a stochastic policy π^* that encodes the updates, which takes existing optimization algorithms as special cases. Our work is related to theirs (cf. Method 2 below and Algorithm 1 in that paper), with the distinction that we include safeguarding and work in the fixed point setting. The idea of Andrychowicz et al. (2016) is to use long short term memory (LSTM) units in recurrent neural networks (RNNs). Additional learning approaches have been applied in the discrete setting (Dai et al., 2018; Li et al., 2018; Bengio et al., 2018).

Our safeguarding scheme is related to existing KM works. Themelis & Patrinos (2019) present a KM method that safeguards in a more hierarchical manner than ours and solely refers to the current iterate residuals (plus a summable sequence). Zhang et al. (2018) use a similar safeguarding step for their Anderson accelerated KM method. However, their methods are not designed to work with L2O.

Our Contribution. Our primary contribution is a simple framework that enables the application of fast data-driven algorithms while maintaining convergence guarantees. This framework can be used with all L2O algorithms that solve convex problems for which proximal and/or gradient oracles are available. We incorporate several safeguarding procedures in a general setting, and we present a simple procedure for utilizing machine learning methods to incorporate knowledge from data sets. These results together form a single, general framework for use by practitioners.

Outline. Section 2 overviews our fixed point setting. Our safeguarded method and results are presented in Section 3. Section 4 describes how to represent and tune L2O algorithms using neural networks. This is followed by numerical examples and conclusions in Sections 5 and 6, respectively.

2. Fixed Point Methods

This section briefly overviews fixed-point methods, which are an abstraction of many optimization algorithms. Let $\langle\cdot,\cdot\rangle$ and $\|\cdot\|$ be the Euclidean inner product and norm defined on $\mathbb{R}^n$, respectively. Denote the set of fixed points of each operator $T:\mathbb{R}^n\to\mathbb{R}^n$ by $\text{Fix}(T):=\{x\in\mathbb{R}^n : Tx=x\}$. For an operator T with a nonempty fixed point set (i.e., $\text{Fix}(T)\neq\emptyset$), we consider the fixed point problem:

$$\text{Find } x^\star \text{ such that } x^\star = T(x^\star). \tag{1}$$

Convex minimization problems, both constrained and unconstrained, may be equivalently rewritten as the problem

(1) for an appropriate operator T. Examples are provided in Section 4 below. We focus on the fixed point formulation to provide a general approach for creating sequences that converge to solutions of (1) and, thus, of the corresponding optimization problem.

The following definitions are used in the sequel. An operator $T : \mathbb{R}^n \to \mathbb{R}^n$ is *nonexpansive* if it is 1-Lipschitz, i.e.,

$$\|Tx - Ty\| \leq \|x - y\|, \quad \text{for all } x, y \in \mathbb{R}^n. \tag{2}$$

An operator T is *averaged* if there exist $\alpha \in (0, 1)$ and a nonexpansive operator $Q : \mathbb{R}^n \to \mathbb{R}^n$ such that $T = (1-\alpha)\mathrm{Id} + \alpha Q$, with Id the identity. The distance $d_C(x)$ between a point $x \in \mathbb{R}^n$ and a set $C \subseteq \mathbb{R}^n$ is given by

$$d_C(x) := \inf\{\|x - y\| : y \in C\}. \tag{3}$$

A classic theorem states that sequences generated by successively applying an averaged operator converge to a fixed point. This method comes from (Krasnosel'skii, 1955) and (Mann, 1953), which yielded adoption of the name *Krasnosel'skiĭ-Mann* (KM) method. This result is stated below and can be found with various forms and proofs in many works (e.g., see (Byrne, 2008, Thm. 5.2) and (Reich, 1979, Thm. 2)).

Theorem 2.1. *If an averaged operator $T : \mathbb{R}^n \to \mathbb{R}^n$ has a nonempty fixed point set and a sequence $\{x^k\}$ with arbitrary initial iterate $x^1 \in \mathbb{R}^n$ satisfies the update relation*

$$x^{k+1} = T(x^k), \quad \textit{for all } k \in \mathbb{N}, \tag{4}$$

then there is a solution $x^\star \in \mathrm{Fix}(T)$ to (1) such that the sequence $\{x^k\}$ converges to $x^\star$, i.e., $x^k \to x^\star$.

In the remainder of this work, we assume each operator T is averaged.

Method 1 Abstract L2O Method (without safeguard)

```
Choose L2O operator $\mathcal{L}_{\mathrm{L2O}}$
Choose parameters $\{\zeta^k\}$                ◁ Take from Training
Choose $x^1 \in \mathbb{R}^n$                  ◁ Initialize iterate
for $k = 1, 2, \dots$ do
    $x^{k+1} \leftarrow \mathcal{L}_{\mathrm{L2O}}(x^k;\ \zeta^k)$      ◁ L2O Update
end for
```

3. Safeguarded L2O Method

This section presents our safeguarded L2O (Safe-L2O) method. Each L2O operator $\mathcal{L}_{\mathrm{L2O}}$ is defined with a parameter ζ. Existing L2O methods may be outlined in an abstract manner by Method 1. Ideally, similar L2O operators are expressed by tuning ζ. In Section 4, we discuss how to choose ζ to yield the "best" update for a particular distribution of data.

In addition to an L2O operator $\mathcal{L}_{\mathrm{L2O}}$, our Safe-L2O method uses a fallback operator T and a scalar sequence $\{\mu_k\}$. Here T defines an averaged operator from the update formula of a conventional optimization algorithm. Each μ_k defines a reference value to determine whether a tentative L2O update is "good." Each reference value μ_k in our proposed safeguarding schemes is relatable to a combination of $\|x^i - T(x^i)\|$ among previous iterates $i = 1, \dots, k$. We illustrate L2O and fallback operators in two examples below.

Example 3.1. Let $f : \mathbb{R}^n \to \mathbb{R}$ be convex and differentiable with L-Lipschitz gradient. Define $\mathcal{L}_{\mathrm{L2O}} : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ by

$$\mathcal{L}_{\mathrm{L2O}}(x;\ \zeta) = x - \frac{2\zeta}{L}\nabla f(x). \tag{5}$$

Here $\mathcal{L}_{\mathrm{L2O}}$ is a gradient descent operator with tunable step-size $2\zeta/L$. It can be shown that $\mathcal{L}_{\mathrm{L2O}}(\cdot\ ;\ \alpha)$ is averaged for $\alpha \in (0, 1)$. Using the gradient descent as the conventional algorithm then implies the fallback operator T can be set to

$$T(x) := \mathcal{L}_{\mathrm{L2O}}\left(x;\ \frac{1}{2}\right) = x - \frac{1}{L}\nabla f(x). \tag{6}$$

Although using $\zeta \gg 1$ may not be theoretically justifiable for L2O updates, this can be useful in accelerating the convergence of a method in some instances (e.g., see (Giselsson et al., 2016)). △

Example 3.2. Let $A \in \mathbb{R}^{m\times n}$, $d \in \mathbb{R}^m$, and define $f(x) := \frac{1}{2}\|Ax - d\|^2$ so that $\nabla f(x) = A^T(Ax - d)$. Next define $\mathcal{L}_{\mathrm{L2O}} : \mathbb{R}^n \times \mathbb{R}^{n\times m} \to \mathbb{R}$ with *matrix* parameter ζ by

$$\mathcal{L}_{\mathrm{L2O}}(x;\ \zeta) := x - \zeta(Ax - d). \tag{7}$$

A fallback operator T can perform gradient descent, i.e.,

$$T(x) := \mathcal{L}_{\mathrm{L2O}}(x;\ \alpha A^t) = x - \alpha A^t(Ax - d), \tag{8}$$

with $\alpha \in (0, 2/\|A^tA\|_2)$. Using various ζ may be more effective for minimizing f (e.g., related to the pseudo inverse of A) than applying T. △

Method 2 Safeguarded L2O (Safe-L2O)

```
Choose L2O operator $\mathcal{L}_{\mathrm{L2O}}$
Choose parameters $\{\zeta^k\}$                       ◁ Take from Training
Choose fallback KM operator $T$
Choose safeguard scheme for $\{\mu_k\}$
Choose $x^1 \in \mathbb{R}^n$, and $\alpha \in [0, 1)$      ◁ Initialize iterate
$\mu_1 \leftarrow \|x^1 - T(x^1)\|$                    ◁ Initialize Safeguard
for $k = 1, 2, \dots$ do
    $y^k \leftarrow \mathcal{L}_{\mathrm{L2O}}(x^k;\ \zeta^k)$              ◁ L2O Prediction
    if $\|y^k - T(y^k)\| \leq \alpha\mu_k$ then        ◁ Safeguard Check
        $x^{k+1} \leftarrow y^k$                        ◁ L2O Update
    else
        $x^{k+1} \leftarrow T(x^k)$                     ◁ Fallback KM Update
    end if
    Compute safeguard update $\mu_{k+1}$             ◁ See Table 1
end for
```

Table 1: Choices to update μ_k that ensure Assumption 2 holds. Here $\alpha, \theta \in (0, 1)$.

NAME	UPDATE FORMULA
Geometric Sequence GS(θ)	$\mu_{k+1} = \begin{cases} \theta\mu_k & \text{if } \|x^{k+1} - T(x^{k+1})\| \leq \alpha\mu_k, \\ \mu_k & \text{otherwise.} \end{cases}$ *Decrease μ_k by geometric factor θ whenever sufficient residual descent occurs.*
Recent Term RT	$\mu_{k+1} = \begin{cases} \|x^{k+1} - T(x^{k+1})\| & \text{if } \|x^{k+1} - T(x^{k+1})\| \leq \alpha\mu_k, \\ \mu_k & \text{otherwise.} \end{cases}$ *Take μ_k to be most recent residual for which sufficient residual descent occurs.*
Arithmetic Average AA	$m_{k+1} := \begin{cases} m_k + 1 & \text{if } \|x^{k+1} - T(x^{k+1})\| \leq \alpha\mu_k, \\ m_k & \text{otherwise,} \end{cases}$ $\mu_{k+1} := \begin{cases} \dfrac{1}{m_k + 1}\left(\|x^{k+1} - T(x^{k+1})\| + m_k\mu_k\right) & \text{if } \|x^{k+1} - T(x^{k+1})\| \leq \alpha\mu_k, \\ \mu_k & \text{otherwise.} \end{cases}$ *Use m_k to count how many times sufficient residual descent occurs and μ_k is the average of the residuals among those times.*
Exponential Moving Average EMA(θ)	$\mu_{k+1} := \begin{cases} \theta\|x^{k+1} - T(x^{k+1})\| + (1-\theta)\mu_{k-1} & \text{if } \|x^{k+1} - T(x^{k+1})\| \leq \alpha\mu_k, \\ \mu_k & \text{otherwise.} \end{cases}$ *Average μ_k with the latest residual whenever sufficient residual descent occurs.*
Recent Max RM(m)	$\Xi_k = \left\{\text{most recent } m \text{ indices } \ell : \|x^\ell - T(x^\ell)\| \leq \alpha\mu_\ell\right\},$ $\mu_{k+1} = \max_{\ell \in \Xi_k} \|x^\ell - T(x^\ell)\|.$ *Take μ_k to be max of the most recent residuals for which sufficient residual descent occurs.*

Our proposed Safe-L2O approach is Method 2. First an L2O operator is chosen in Line 1. For each iteration k, the parameter ζ^k is used to define the L2O update $\mathcal{L}_{\text{L2O}}(\cdot\,;\,\zeta^k)$. These parameters are chosen in Line 2 (e.g., following standard training methods for machine learning models as outlined by Procedure 3). A fallback operator is chosen in Line 3 (e.g., using Table 2). The scheme for each safeguard parameter μ_k is chosen in Line 4 (e.g., using Table 1). Next the initial iterate x^1 and a weighting term α are chosen in Line 5. The safeguard sequence initial iterate is then initialized using the initial iterate x^1 and the fallback operator T in Line 6. From Line 7 to Line 14, a repeated loop occurs to compute each update x^{k+1}. In Line 8 the L2O operator is applied to the current iterate x^k get a tentative update y^k. This y^k is then determined to be "good" if the the inequality in Line 9 holds. In such a case, the L2O update is assigned to x^{k+1} in Line 10. Otherwise, the fallback operator T is used to obtain the update x^{k+1} in Line 12. Lastly, the safeguard parameter is updated in Line 14.

Below are several standard assumptions used to prove our convergence result in Theorem 3.1.

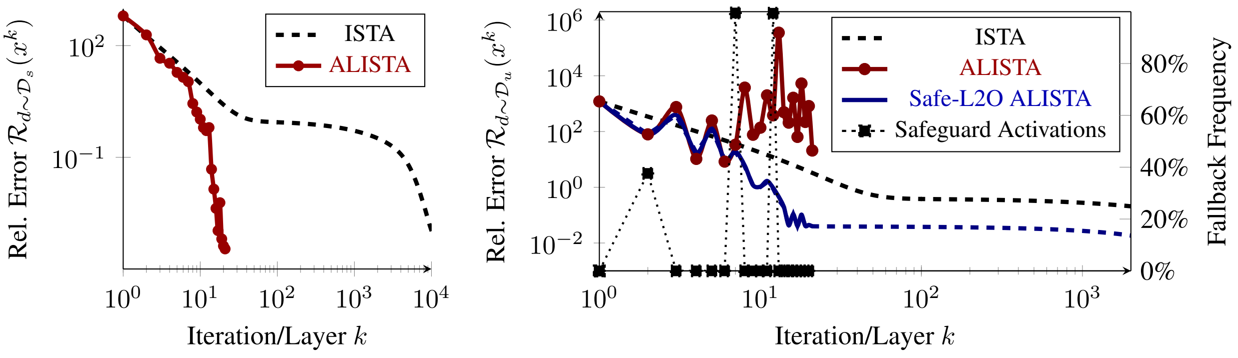
Assumption 1. The underlying optimization problem has a solution and is equipped with an operator T satisfying:

1. $\text{Fix}(T)$ equals the solution set (so $\text{Fix}(T) \neq \emptyset$),
2. T is averaged,
3. $\text{Id} - T$ is coercive, i.e.,

$$\lim_{\|x\| \to \infty} \|x - T(x)\| = \infty. \tag{9}$$

Remark 3.1. Assumption 1 Part 3 does not always hold, but it does for a minor perturbation. Suppose f is a smooth function with ∇f bounded along a sequence $\{x^k\}$ for which $\|x^k\| \to \infty$. Define the gradient operator $T(x) := x - \alpha\nabla f(x)$. Then $\{x^k - T(x^k)\} = \{\alpha\nabla f(x^k)\}$. Fixing $\varepsilon > 0$ and setting $\tilde{f}(x) := f(x) + \frac{\varepsilon}{2}\|x\|^2$ yields an associated $\tilde{T}$ for which $\{x^k - \tilde{T}(x^k)\} = \{\alpha\nabla f(x^k) + \varepsilon x^k\}$, which blows up as $k \to \infty$. Since this works for small ε, in practice it may be reasonable to assume Part 3 holds for Safe-L2O.

The next assumption ensures the used L2O updates approach the solution set. This is accomplished computing the fixed point residual with the fallback operator.



(a) Performance on seen distribution, i.e., $d \sim \mathcal{D}_s$

(b) Performance on *unseen* distribution, i.e., $d \sim \mathcal{D}_u$

Figure 1: Plot of error versus iteration for ALISTA example. Trained with $\phi_d = f_d$. Inferences used $\alpha = 0.99$. In (b), the safeguard is EMA(0.25) and how often the L2O update is "bad" and the safeguard activates for Safe-L2O is indicated in reference to the right vertical axis. This plot shows the safeguard is used only when $k = 2$, $k = 7$, and $k = 12$.

Assumption 2. If the inequality $\|y^k - T(y^k)\| \leq \alpha\mu_k$ holds infinitely many times, then $\{\mu_k\}$ converges to zero. ◇

We clarify that we are *not* assuming the inequality holds infinitely many times. We are merely specifying how $\{\mu_k\}$ should behave. In fact, the inequality holds only when an L2O update is good, and so it may hold just a few times.

Our proposed methods for choosing the sequence $\{\mu_k\}$ that we will show satisfying Assumption 2 are in Table 1. These methods are *adaptive* in the sense that each update to μ_k depends upon the iterate x^k and (possibly) previous iterates. Each safeguard parameter μ_k also remains constant in k except for when the residual norm $\|x^{k+1} - T(x^{k+1})\|$ decreases to less than a geometric factor of μ_k. This allows each μ_k to trail the value of the residual norm $\|x^k - T(x^k)\|$ and allows the residual norm to increase in k from time to time. As noted in the introduction, this trailing behavior provides flexibility to the L2O updates. Our main result is below and it is followed by a corollary justifying use of the schemes in Table 1 (both proven in the Appendix).

Theorem 3.1. *If $\{x^k\}$ is a sequence generated by the Safe-L2O method and Assumptions 1 and 2 hold, then*

$$\lim_{k\to\infty} d_{\mathrm{Fix}(T)}\left(x^k\right) = 0, \tag{10}$$

that is, the sequence approaches the solution set. And, if $\{x^k\}$ contains a single cluster point, then $\{x^k\}$ converges to a point $x^\star \in \mathrm{Fix}(T)$ that is a solution to (1).

Corollary 3.1. *If $\{x^k\}$ is a sequence generated by the Safe-L2O method and Assumption 1 holds and $\{\mu_k\}$ is generated using a scheme outlined in Table 1, then Assumption 2 holds and, by Theorem 3.1, the limit (31) holds.*

Remark 3.2. The above results hold even in the Hilbert space setting (with weak convergence if the space has an infinite dimension) and when Assumption 1 Part 3 is weakened to instead assume $\liminf_{\|x\|\to\infty} \|x - T(x)\| > 0$.

Table 3: Training loss function ϕ_d choices. The fallback is T_d and $x_d^\star$ and $z_d^\star$ are primal and dual solutions, respectively.

Problem	Loss Function ϕ_d
$\min f_d(x)$	f_d
$\min f_d(x)$	$\|x - x_d^\star\|^2$
$\min f_d(x)$	$\|x - T_d(x)\|^2$
$\min f_d(x)$ s.t. $Ax = d$	$\|x - T_d(x)\|^2$
$\min f_d(x)$ s.t. $Ax = d$	$\|x - x_d^\star\|^2 + \|z - z_d^\star\|^2$

4. Training and Averaged Operator Selection

Safe-L2O may be executed via inferences of a feed forward neural network. The input into the network is the data d, often in vector form. Each layer is designed so that its input is x^k, to which it applies either an L2O or fallback update (following the Safe-L2O method), and outputs x^{k+1} to the next layer. We encode all the network parameters with $\Theta := \{\zeta^k\}$. The set over which Θ is minimized, may be chosen with great flexibility. If each parameter ζ^k may be chosen independently, the network weights vary by layer. This is used in our numerical examples below. If instead each ζ^k is identical, i.e., the parameters across all layers are fixed, then we obtain a recurrent neural network (RNN). For each application of the algorithm, the fallback operator changes, depending upon the data d. To make explicit this dependence, we henceforth add a subscript to write T_d.

The "optimal" choice of parameters Θ depends upon the application. Suppose each d is drawn from a common distribution $\mathcal{D}$. Then a choice of "optimal" parameters $\Theta^\star$ may be identified as those for which the expected value of $\phi_d(x^K)$ is minimized among $d \sim \mathcal{D}$, where $\phi_d : \mathbb{R}^n \to \mathbb{R}$

Table 2: Averaged operators for well-known algorithms. We assume $\alpha > 0$ and, when α is multiplied by a gradient, we also assume $\alpha < 2/L$, with L the Lipschitz constant for the gradient. The dual of a function is denoted by a superscript $*$ and $\Omega := \{(x,z) : Ax + Bz = b\}$. Operators J and R are defined in equations (12) and (13), respectively. The block matrix M is $M = [\alpha^{-1}\text{Id}, A^T; -A, \beta^{-1}\text{Id}]$. In each case, $\mathcal{L}$ is the Lagrangian associated with the presented problem.

Problem	Method	Fallback Operator T
$\min f(x)$	Gradient Descent	$\text{Id} - \alpha\nabla f$
$\min f(x)$	Proximal Point	$\text{prox}_{\alpha f}$
$\min\{g(x) : x \in C\}$	Projected Gradient	$\text{proj}_C \circ (\text{Id} - \alpha\nabla g)$
$\min f(x) + g(x)$	Proximal Gradient	$\text{prox}_{\alpha f} \circ (\text{Id} - \alpha\nabla g)$
$\min f(x) + g(x)$	Douglas-Rachford	$\frac{1}{2}\left(\text{Id} + R_{\alpha\partial f} \circ R_{\alpha\partial g}\right)$
$\min\limits_{(x,z)\in\Omega} f(x) + g(z)$	ADMM	$\frac{1}{2}\left(\text{Id} + R_{\alpha A\partial f^*(A^T\cdot)} \circ R_{\alpha(B\partial g^*(B^T\cdot)-b)}\right)$
$\min f(x)$ s.t. $Ax = b$	Uzawa	$\text{Id} + \alpha\left(A\nabla f^*(-A^T\cdot) - b\right)$
$\min f(x)$ s.t. $Ax = b$	Proximal Method of Multipliers	$J_{\alpha\partial\mathcal{L}}$
$\min f(x) + g(Ax)$	PDHG	$J_{M^{-1}\partial\mathcal{L}}$

is an appropriate cost function (e.g., as in Table 3). This is expressed mathematically by stating $\Theta^\star$ solves the problem

$$\min_{\Theta} \mathbb{E}_{d\sim\mathcal{D}}\left[\phi_d(x^K(\Theta,d))\right], \tag{11}$$

where we emphasize the dependence of x^K on Θ and d by writing $x^K = x^K(\Theta,d)$. We approximately solve the problem (11) by sampling data $\{d^n\}_{n=1}^N$ from $\mathcal{D}$ and minimizing an empirical loss function. A summary for training is outlined in Procedure 3. Note different learning problems than (11) may be used (e.g., the min-max problem used by adversarial networks (Goodfellow et al., 2014)).

We provide the following note about proximals and resolvents for reference when identifying T_d from conventional algorithms. Consider a convex function $f : \mathbb{R}^n \to \mathbb{R}$ with subgradient ∂f. For $\alpha > 0$, the *resolvent* of $\alpha\partial f$ is

$$\begin{aligned} J_{\alpha\partial f}(x) &:= (\text{Id} + \alpha\partial f)^{-1}(x) \\ &= \left\{ y : \frac{x-y}{\alpha} \in \partial f(y) \right\}, \end{aligned} \tag{12}$$

and the *reflected resolvent* of ∂f is

$$R_{\alpha\partial f}(x) := (2J_{\alpha\partial f} - \text{Id})(x) = 2J_{\alpha\partial f}(x) - x. \tag{13}$$

If f is closed, convex, and proper, then the resolvent is precisely the proximal operator, i.e.,

$$J_{\alpha\partial f} = \text{prox}_{\alpha f}(x) := \operatorname*{arg\,min}_{z\in\mathbb{R}^n} \alpha f(z) + \frac{1}{2}\|z - x\|^2. \tag{14}$$

Proximal operators for several well-known functions can be expressed by explicit formulas (e.g., see page 177 in (Beck, 2017)). It can be shown that $R_{\alpha\partial f}$ is nonexpansive and $J_{\alpha\partial f}$ is averaged (e.g., see Prop. 4.4, Thm. 20.25, Example 23.3, and Prop. 23.8 in (Bauschke & Combettes, 2017)). Table 2 provides several examples of the use of these operators in well-known optimization algorithms.

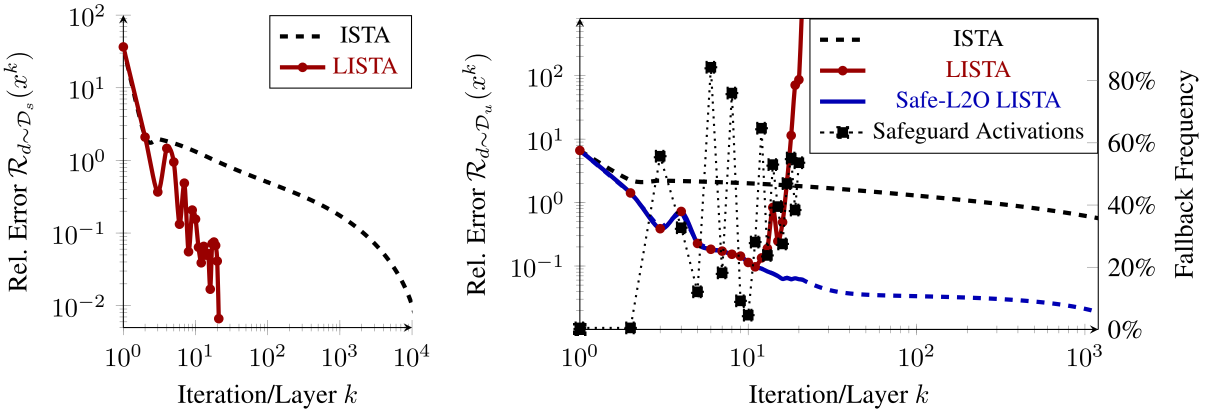
Procedure 3 How to Tune L2O Parameters $\{\zeta^k\}$

```
1: Choose operator $\mathcal{L}_{\text{L2O}}$
2: Choose number of iterations/layers $K$ for training
3: Initialize weights $\Theta = \{\zeta^k\}$
4: Choose training data distribution $\mathcal{D}$
5: Choose training loss function $\phi_d$
6: Compute "optimal" weights
       $\Theta^\star \in \arg\min_{\Theta} \mathbb{E}_{d\sim\mathcal{D}}\left[\phi_d(x^K)\right]$,
   with $x^K$ being the $K$-th iterate of Method 1
```

5. Numerical Examples

This section presents examples using Safe-L2O.[1] We numerically investigate (i) the convergence rate of Safe-L2O relative to corresponding conventional algorithms, (ii) the efficacy of safeguarding procedures when inferences are performed on data for which L2O fails intermittently, and (iii) the convergence of Safe-L2O schemes even when the application of $\mathcal{L}_{\text{L2O}}$ is not justified theoretically. We first use $\mathcal{L}_{\text{L2O}}$ from ALISTA (Liu et al., 2019a) on a synthetic

[1] All of the codes for this work can be found on GitHub here: (link will be added after review process).



(a) Performance on seen distribution $\mathcal{D}_s$

(b) Performance on *unseen* distribution, i.e., $d \sim \mathcal{D}_u$

Figure 2: Plot of error versus iteration for LISTA denoising. Trained with $\phi_d = f_d$. Inferences used $\alpha = 0.99$ and EMA(0.25). In (b), how often the L2O update is "bad" and the safeguard activates for Safe-L2O is indicated in reference to the right vertical axis. This plot shows the safeguard is used intermittently for $k > 2$.

LASSO problem. We then use LISTA on a LASSO problem for image processing, differentiable linearized ADMM (Xie et al., 2019) on a sparse coding problem, and a simple L2O method for nonnegative least squares (NNLS).

In each example, $f_d^\star$ denotes the optimal value of $f_d(x)$ among all possible x. Performance is measured using a modified relative objective error:

$$\mathcal{R}_{f,\mathcal{D}}(x) := \frac{\mathbb{E}_{d\sim\mathcal{D}}[f_d(x) - f_d^\star]}{\mathbb{E}_{d\sim\mathcal{D}}[f_d^\star]}, \tag{15}$$

where the expectations are estimated numerically (see Appendix for details). We use (15) rather than the expectation of relative error to avoid high sensitivity to outliers.

Our numerical results are presented in several plots. When each iterate x^k is computed using data d drawn from the same distribution $\mathcal{D}_s$ that was used to train the L2O algorithm, we say the performance is on the "seen" distribution $\mathcal{D}_s$. These plots form the primary illustrations of the speedup of L2O algorithms. When each d is drawn from a distribution $\mathcal{D}_u$ that is *different* than $\mathcal{D}_s$, we refer to $\mathcal{D}_u$ as the *unseen* distribution. These plots show the ability of the safeguard to ensure convergence. A dotted plot with square markers is also added to show the frequency of safeguard activations among test samples. We extend the Safe-L2O methods beyond their training iterations by applying the fallback operator T; we demarcate where this extension begins by changing the Safe-L2O plots from solid to dashed lines.

5.1. ALISTA for LASSO

Here we consider the LASSO problem for sparse coding. Let $x^\star \in \mathbb{R}^{500}$ be a sparse vector and $A \in \mathbb{R}^{250\times 500}$ be a dictionary. We assume access is given to noisy linear measurements $d \in \mathbb{R}^{250}$, where $\varepsilon \in \mathbb{R}^{250}$ is additive Gaussian white noise and $d = Ax^\star + \varepsilon$. Even for underdetermined systems, when $x^\star$ is sufficiently sparse and $\tau \in (0, \infty)$ is an appropriately chosen regularization parameter, $x^\star$ can often be recovered reasonably by solving the LASSO problem

$$\min_{x\in\mathbb{R}^n} f_d(x) := \frac{1}{2}\|Ax - d\|_2^2 + \tau\|x\|_1, \tag{16}$$

where $\|\cdot\|_2$ and $\|\cdot\|_1$ are the ℓ_2 and ℓ_1 norms, respectively. A classic method for solving (16) is the iterative shrinkage thresholding algorithm (ISTA) (e.g., see (Daubechies et al., 2004)).[2] Liu et al. (2019a) present the L2O scheme ALISTA that we implement here. This L2O operator $\mathcal{L}_{\text{L2O}}$ is parameterized by $\zeta = (\theta, \gamma) \in \mathbb{R}^2$. Further implementation details for the LASSO problem may be found in the Appendix.

5.2. Linearized ADMM

Let $A \in \mathbb{R}^{250\times 500}$ and $d \in \mathbb{R}^{250}$ be as in Subsection 5.1. Here we apply the L2O scheme differentiable linearized ADMM (D-LADMM) of Xie et al. (2019) to the closely related sparse coding problem

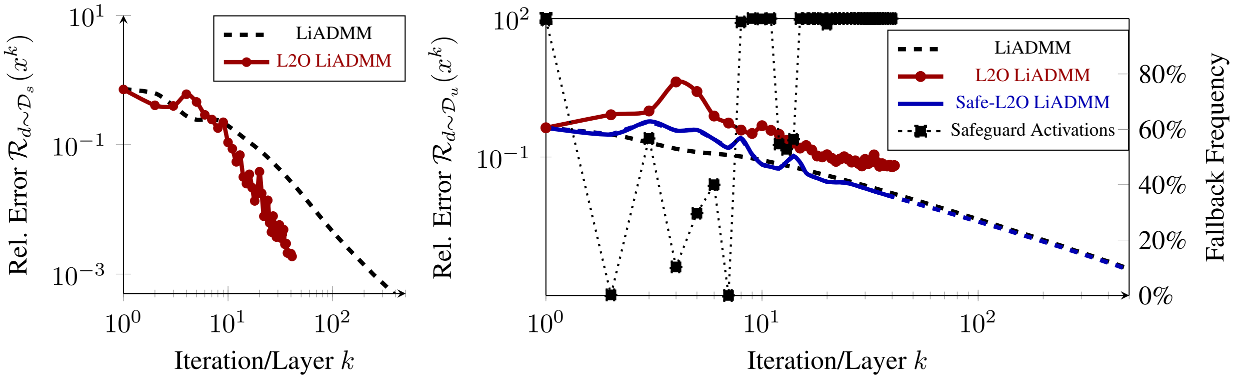
$$\min_{x\in\mathbb{R}^n} \|Ax - d\|_1 + \tau\|x\|_1. \tag{17}$$

The L2O operator $\mathcal{L}_{\text{L2O}}$ and fallback linearized ADMM (LiADMM) operator T are provided in the Appendix along with implementation details. Plots are provided in Figure 3.

5.3. LISTA for Natural Image Denoising

To evaluate our safeguarding mechanism in a more realistic setting, we apply safeguarded LISTA to a natural image denoising problem. In this subsection, we learn a LISTA-CP

[2] This is a special case of the proximal-gradient in Table 2.



(a) Performance on seen distribution $\mathcal{D}_s$

(b) Performance on *unseen* distribution, i.e., $d \sim \mathcal{D}_u$

Figure 3: Plot of error versus iteration for D-LADMM. Trained with $\phi_d = f_d$. Inferences used $\alpha = 0.99$ and EMA(0.75). In (b), how often the L2O update is “bad” and the safeguard activates for Safe-L2O is indicated in reference to the right vertical axis. This plot shows the safeguard is used about 50% and 0% of the time when $k = 3$ and $k = 7$, respectively.

model (Chen et al., 2018) to perform natural image denoising. During training, L2O LISTA-CP model is trained to recover clean images from their Gaussian noisy counterparts by solving (16). In (16), d is the noisy input to the model, and the clean image is recovered with $\hat{d} = Ax^\star$, where $x^\star$ is the optimal solution. The dictionary $A \in \mathbb{R}^{256\times 512}$ is learned on the BSD500 dataset (Martin et al., 2001) by solving a dictionary learning problem (Xu & Yin, 2014). During testing, however, the learned L2O LISTA-CP is applied to unseen pepper-and-salt noisy images. Comparison plots are provided in Figure 2 and visualized results are shown in Appendix along with implementation details.

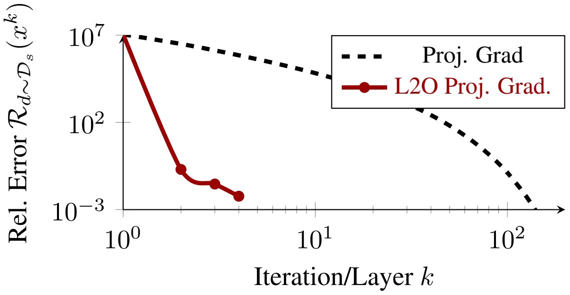


Figure 4: Performance on seen distribution $\mathcal{D}_s$ of the L2O projected gradient scheme in (19)

5.4. Projected Gradient for Nonnegative Least Squares

Let $A \in \mathbb{R}^{500\times 250}$ and $d \in \mathbb{R}^{500}$. Here we consider an overdetermined NNLS problem

$$\min_{x\in\mathbb{R}^{250}} f_d(x) := \frac{1}{2}\|Ax - d\|_2^2 \quad \text{s.t.} \quad x \geq 0. \tag{18}$$

Generalizing the projected-gradient method, we use

$$\mathcal{L}_{\text{L2O}}(x;\, \zeta) := \max\left(x - \zeta(Ax - d), 0\right), \tag{19}$$

where $\zeta \in \mathbb{R}^{250\times 500}$. The fallback method is projected gradient, i.e., $T(x) := \mathcal{L}_{\text{L2O}}(x;\, \alpha A^T)$ where $\alpha = 1/\|A^t A\|_2$. Here $\Theta = (\zeta^k)_{k=1}^K$ consists of mnK trainable parameters. A summary plot is given in Figure 4. Since this problem is unregularized, the L2O method learned very efficient updates, given A. This resulted in comparable performance on unseen data and the safeguard was never activated. Further implementation details may be found in the Appendix.

6. Conclusions

Numerous insights may be drawn from our examples. The first observation is that, roughly speaking, ALISTA, LISTA, and the L2O method for NNLS all reduced computational costs by multiple orders of magnitude when applied to data from the same distribution as the training data. This is expected, given the results of previous works. More importantly, plots (b) in Figures 1 to 3 demonstrate that the safeguard steers updates to convergence when they would otherwise diverge or converge slower than the conventional algorithm. Those plots show the convergence of Safe-L2O on data distinct from training and the divergence of the nonsafeguarded L2O schemes.

This work proposes a framework for ensuring convergence of L2O algorithms. Sequences generated by our Safe-L2O method provably approach the solution set. When there is a unique cluster point, this yields convergence to a solution. Our Safe-L2O algorithm is also easy to implement using neural networks. Our numerical experiments demonstrate rapid convergence by Safe-L2O methods and effective safeguarding when the L2O schemes appear to otherwise diverge. Future work will provide a better data-driven fallback method and investigate stochastic extensions.

References

Ablin, P., Moreau, T., Massias, M., and Gramfort, A. Learning step sizes for unfolded sparse coding. *arXiv:1905.11071*, 2019.

Andrychowicz, M., Denil, M., Gómez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and de Freitas, N. Learning to learn by gradient descent by gradient descent. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 3981–3989. Curran Associates, Inc., 2016.

Bauschke, H. and Combettes, P. *Convex Analysis and Monotone Operator Theory in Hilbert Spaace*. Springer, 2nd. edition, 2017.

Bauschke, H. H., Iorio, F., and Koch, V. R. The Method of Cyclic Intrepid Projections: Convergence Analysis and Numerical Experiments. In Wakayama, M., Anderssen, R. S., Cheng, J., Fukumoto, Y., McKibbin, R., Polthier, K., Takagi, T., and Toh, K.-C. (eds.), *The Impact of Applications on Mathematics*, volume 1, pp. 187–200. Springer Japan, Tokyo, 2014.

Beck, A. *First-order methods in optimization*, volume 25. SIAM, 2017.

Bengio, Y., Lodi, A., and Prouvost, A. Machine learning for combinatorial optimization: a methodological tour d'horizon. *arXiv:1811.06128*, 2018.

Blumensath, T. and Davies, M. E. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3):265–274, 2009. ISSN 1063-5203.

Borgerding, M., Schniter, P., and Rangan, S. AMP-Inspired Deep Networks for Sparse Linear Inverse Problems. *IEEE Transactions on Signal Processing*, 65(16):4293–4308, 2017.

Byrne, C. L. *Applied Iterative Methods*. A K Peters, Ltd., 2008.

Cegielski, A. *Iterative Methods for Fixed Point Problems in Hilbert Spaces*. Number 2057 in Lecture Notes in Mathematics. Springer, 2012.

Chen, X., Liu, J., Wang, Z., and Yin, W. Theoretical Linear Convergence of Unfolded ISTA and Its Practical Weights and Thresholds. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 9061–9071. Curran Associates, Inc., 2018.

Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., and Song, L. Learning Combinatorial Optimization Algorithms over Graphs. *arXiv:1704.01665*, February 2018.

Daubechies, I., Defrise, M., and Mol, C. D. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

Giryes, R., Eldar, Y. C., Bronstein, A. M., and Sapiro, G. Tradeoffs Between Convergence Speed and Reconstruction Accuracy in Inverse Problems. *IEEE Transactions on Signal Processing*, 66(7):1676–1690, 2018.

Giselsson, P., Falt, M., and Boyd, S. Line search for averaged operator iteration. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 1015–1022, Las Vegas, NV, USA, December 2016. IEEE.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Gregor, K. and LeCun, Y. Learning Fast Approximations of Sparse Coding. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pp. 399–406, USA, 2010. Omnipress.

Groetsch, C. A note on segmenting Mann iterates. *Journal of Mathematical Analysis and Applications*, 40(2):369–372, November 1972.

Hershey, J. R., Roux, J. L., and Weninger, F. Deep Unfolding: Model-Based Inspiration of Novel Deep Architectures. *arXiv:1409.2574*, 2014.

Krasnosel'skii, M. Two remarks about the method of successive approximations. *Uspekhi Mat. Nauk*, 10:123–127, 1955.

Li, K. and Malik, J. Learning to Optimize. *arXiv:1606.01885*, June 2016.

Li, Z., Chen, Q., and Koltun, V. Combinatorial optimization with graph convolutional networks and guided tree search. In *Advances in Neural Information Processing Systems*, pp. 539–548, 2018.

Liu, J., Chen, X., Wang, Z., and Yin, W. ALISTA: Analytic weights are as good as learned weights in LISTA. In *International Conference on Learning Representations*, 2019a.

Liu, Q., Shen, X., and Gu, Y. Linearized ADMM for Nonconvex Nonsmooth Optimization With Convergence Analysis. *IEEE Access*, 7:76131–76144, 2019b.

Mann, R. Mean Value Methods in Iteration. 4(3):506–510, 1953.

Martin, D., Fowlkes, C., Tal, D., and Malik, J. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pp. 416–423, July 2001.

Metzler, C., Mousavi, A., and Baraniuk, R. Learned D-AMP: Principled Neural Network based Compressive Image Recovery. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 1772–1783. Curran Associates, Inc., 2017.

Moreau, T. and Bruna, J. Understanding Trainable Sparse Coding with Matrix Factorization. 2017.

Reich, S. Weak convergence theorems for nonexpansive mappings in Banach spaces. *Journal of Mathematical Analysis and Applications*, 67(2):274–276, 1979.

Sprechmann, P., Bronstein, A. M., and Sapiro, G. Learning Efficient Sparse and Low Rank Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37 (9):1821–1833, September 2015.

Themelis, A. and Patrinos, P. SuperMann: a superlinearly convergent algorithm for finding fixed points of nonexpansive operators. 2019.

Wang, Z., Chang, S., Zhou, J., Wang, M., and Huang, T. Learning A Task-Specific Deep Architecture For Clustering. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, Proceedings, pp. 369–377. Society for Industrial and Applied Mathematics, June 2016a.

Wang, Z., Ling, Q., and Huang, T. S. Learning Deep ℓ_0 Encoders. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016b.

Wang, Z., Liu, D., Chang, S., Ling, Q., Yang, Y., and Huang, T. S. D3: Deep Dual-Domain Based Fast Restoration of JPEG-Compressed Images. pp. 2764–2772, 2016c.

Wang, Z., Yang, Y., Chang, S., Ling, Q., and Huang, T. S. Learning a Deep l_∞ Encoder for Hashing. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pp. 2174–2180. AAAI Press, 2016d.

Xie, X., Wu, J., Zhong, Z., Liu, G., and Lin, Z. Differentiable Linearized ADMM. *arXiv:1905.06179*, 2019.

Xin, B., Wang, Y., Gao, W., Wipf, D., and Wang, B. Maximal Sparsity with Deep Networks? In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 4340–4348. Curran Associates, Inc., 2016.

Xu, Y. and Yin, W. A fast patch-dictionary method for whole image recovery. *arXiv preprint arXiv:1408.3740*, 2014.

Yang, Y., Sun, J., Li, H., and Xu, Z. Deep ADMM-Net for Compressive Sensing MRI. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 10–18. Curran Associates, Inc., 2016.

Zhang, J., O'Donoghue, B., and Boyd, S. Globally convergent type-i anderson acceleration for non-smooth fixed-point iterations. *arXiv:1808.03971*, 2018.

A. Appendix

The Appendix contains a subsection with supplemental materials for the numerical examples along with a subsection with proofs of the theoretical results.

A.1. Numerical Example Supplement Materials

We begin with a general note on training and then consider individual experiments. The training procedure for each experiment was conducted layerwise. By this, we mean that first the network weights were tuned using one layer. Then the network was trained with two layers, using the learned weights from the first layer as an initialization/warm start for that layer's parameters. This was then repeated until the final number K of layers was reached. This approach is built upon the intuition that, because the network layers model an optimization algorithm that progressively improves, each successive layer's weights likely depend upon the previous layers weights.

Supplement to Subsection 5.1. In similar manner to (Chen et al., 2018) and (Liu et al., 2019a), we use the following setup. We take $m = 250$, $n = 500$, and $\tau = 0.001$. Each entry of the dictionary A is sampled i.i.d from the standard Gaussian distribution, i.e., $a_{ij} \sim \mathcal{N}(0, 1/m)$. Having these entries, we then normalize each column of A, with respect to the Euclidean norm. Each d in the distribution $\mathcal{D}_s$ of data used to train the neural network is constructed using $d = Ax^\star + \varepsilon$ with noise $\varepsilon \sim 0.1 \cdot \mathcal{N}(0, 1/m)$ and each entry of $x^\star$ as the composition of Bernoulli and Gaussian distributions, i.e., $x^\star_j \sim \text{Ber}(0.1) \circ \mathcal{N}(0,1)$ for all $j \in [n]$. Each d in the *unseen* distribution $\mathcal{D}_u$ is computed using the same distribution of noise ε as before and using $x^\star_j \sim \text{Ber}(0.2) \circ \mathcal{N}(0, 2)$. Our data set consists of 10,000 training samples and 1,000 test samples.

Given $x^1 \in \mathbb{R}^n$, the ISTA method iteratively computes

$$x^{k+1} := T(x^k) := \eta_{\tau/L}\left(x^k - \frac{1}{L}A^T(Ax^k - d)\right), \quad (20)$$

where $L = \|A^t A\|_2$ and η_θ is the soft-thresholding function defined by component-wise operations:

$$\eta_\theta(x) := \text{sgn}(x) \cdot \max\{0, |x| - \theta\}. \quad (21)$$

We applied Safe-L2O to the LASSO problem above by using T defined in (20) and the $\mathcal{L}_{\text{L2O}}$ update operation from ALISTA (Liu et al., 2019a). The L2O operator $\mathcal{L}_{\text{L2O}}$ is parameterized by $\zeta = (\theta, \gamma)$ for positive scalars θ and γ and defined by

$$\mathcal{L}_{\text{L2O}}(x;\ \zeta) := \eta_\theta\left(x - \gamma W^T(Ax - d)\right), \quad (22)$$

where

$$W \in \operatorname*{arg\,min}_{M \in \mathbb{R}^{m\times n}} \|M^T A\|_F, \quad \text{subject to } (M_{:,\ell})^T A_{:,\ell} = 1, \ \text{ for all } \ell \in [n], \quad (23)$$

and $\|\cdot\|_F$ is the Frobenius norm and the Matlab notation $M_{:,\ell}$ is used to denote the ℓth column of the matrix M. The parameter $\Theta(\theta^k, \gamma^k)_{k=1}^K$ consists of $2K$ scalars.

Supplement to Subsection 5.2. LADMM is used to solve problems of the form

$$\min_{x\in\mathbb{R}^n, z\in\mathbb{R}^m} f(x) + g(z) \quad \text{s.t.} \quad Ax + Bz = d, \quad (24)$$

for which LADMM generates sequences $\{x^k\}$, $\{z^k\}$ and $\{u^k\}$ defined by the updates

$$\begin{aligned} x^{k+1} &:= \text{prox}_{\beta f}\left(x^k - \beta A^T\left[u^k + \alpha\left(Ax^k + Bz^k - d\right)\right]\right), \\ z^{k+1} &:= \text{prox}_{\gamma g}\left(z^k - \gamma B^T\left[u^k + \alpha\left(Ax^{k+1} + Bz^k - d\right)\right]\right), \\ u^{k+1} &:= u^k + \alpha\left(Ax^{k+1} + Bz^{k+1} - d\right), \end{aligned} \quad (25)$$

with given scalars $\alpha, \beta, \gamma \in (0, \infty)$. The problem (17) may be written in the form of (24) by taking $f = \tau\|\cdot\|_1$, $g = \|\cdot\|_1$, and $B = -\text{Id}$. In this case, the proximal operators in (25) reduce to soft-thresholding operators. Although not given in Table 2, at each iteration of LADMM there is an associated iterate ν^k for which the update ν^{k+1} is generated by applying an averaged operator T to the current iterate ν^k. As shown in Lemma 1 of Subsection A.2, for our setup, assuming $\gamma = 1/\alpha$ and $\alpha\beta\|A^T A\|_2 < 1$, the norm of the associated fixed point residual at the iterate ν^k is given by

$$\|\nu^k - T(\nu^k)\| = \left\|\begin{bmatrix} Ax^{k+2} - z^{k+1} - d \\ P(x^{k+2} - x^{k+1}) \end{bmatrix}\right\|, \quad (26)$$

with P defined below in (73). For notational clarity, the term x^k in the SKM and LSKM schemes is replaced in this subsection by the tuple (x^k, z^k, u^k). This is of practical importance too since it is the sequence $\{x^k\}$ that converges to a solution of (17).

We now modify the iteration (25) for the problem (17) to create the D-LADMM L2O scheme. We generalize soft-thresholding to vectorized soft-thresholding for $\beta \in \mathbb{R}^n$ by

$$\eta_\beta(x) = (\eta_{\beta_1}(x_1),\ \eta_{\beta_2}(x_2),\ \ldots,\ \eta_{\beta_n}(x_n)). \quad (27)$$

We assume η_β represents the scalar soft-thresholding in (21) when $\beta \in \mathbb{R}$ and the vector generalization (27) when $\beta \in \mathbb{R}^n$. Combining ideas from ALISTA (Liu et al., 2019a) and D-LADMM (Liu et al., 2019b), given $(x^k, z^k, \nu^k) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^m$, $\alpha^k, \gamma^k, \xi^k \in \mathbb{R}^m$, $\beta^k, \sigma^k \in \mathbb{R}^n$, $W_1 \in$

$\mathbb{R}^{n\times m}$, and $W_2 \in \mathbb{R}^{m\times m}$, set

$$\begin{aligned}
\tilde{x}^{k+1} &:= \eta_{\beta^k}\left(x^k - \sigma^k \circ (W_1^k)^T\left[\nu^k + \alpha_k \circ \left(Ax^k - z^k - d\right)\right]\right),\\
\tilde{z}^{k+1} &:= \eta_{\gamma^k}\left(z^k - \xi^k \circ (W_2^k)^T\left[\nu^k + \alpha_k \circ \left(A\tilde{x}^{k+1} - z^k - d\right)\right]\right),\\
\tilde{\nu}^{k+1} &:= \nu^k + \alpha_k \circ \left(A\tilde{x}^{k+1} - \tilde{z}^{k+1} - d\right),
\end{aligned} \tag{28}$$

with element-wise products denoted by $\circ$. For the parameter $\zeta^k := (\alpha^k, \beta^k, \gamma^k, \sigma^k, \xi^k, W_1^k, W_2^k)$, then define

$$\mathcal{L}_{\mathrm{L2O}}(x^k, z^k, \nu^k;\, \zeta^k) := (\tilde{x}^{k+1}, \tilde{z}^{k+1}, \tilde{\nu}^{k+1}). \tag{29}$$

Fixing the number of iterations K, the learnable parameters from (28) used in the LSKM Algorithm may be encoded by $\Theta = (\zeta^k)_{k=1}^K = \left(\alpha^k, \beta^k, \gamma^k, \sigma^k, \xi^k, W_1^k, W_2^k\right)_{k=1}^K$, consisting of $(2n+3m+mn+m^2)K$ scalars. To stabilize the training process, we share the W_1 across all layers in practice. We also fix $W_2 = -\mathrm{Id}$ and only learn the step sizes ξ^k before it. Moreover, different from other experiments, we add an additional end-to-end training stage after we finish the normal layerwise training described at the beginning of the Appendix, which is found helpful to improve the recovery performance at the final output layer.

For data generation, we use the same settings for the dictioanry A and sparse vectors x^* as in the experiments of Subsection 5.1. But we make a small modification to to generation of noise ε due to the ℓ_1-ℓ_1 objective, where we sample ε from the same (seen and unseen) distribution as x^*, i.e. the noises are also sparse. We choose $\tau = 1.0$.[3]

Supplement to Subsection 5.3. We choose LISTA with coupled weight (i.e. LISTA-CP) in (Chen et al., 2018) for natural image denoising. This is done for two reasons: 1) LISTA-CP has a larger capacity to work well in complex real-world settings; 2) the dictionary A learned from natural images is much more ill-conditioned than the Gaussian matrix in Subsection 5.1. The dictionary $A \in \mathbb{R}^{256\times 512}$ is learned by solving a dictionary learning problem (Xu & Yin, 2014) on 16×16 image patches extracted from BSD500 dataset (Martin et al., 2001) [4].

The training set that we use includes 50,000 images patches of size 16×16 randomly extracted from 200 images in the BSD500 training set, 250 patches each. White Gaussian noises with standard deviation $\sigma = 30$ (pixel values range from 0 to 255) are added to the patches. For testing, we use the "Peppers" image as the ground truth, which is 1,024 non-overlapping patches. The noisy testing patches in the *seen* distribution is generated in the same way as the training set. The testing patches in the *unseen* distribution is polluted by *pepper-and-salt* noises with density $r = 70\%$.

The update operation from LISTA-CP (Chen et al., 2018) is similar to (22) but has one more matrix weight to learn in each layer:

$$\mathcal{L}_{\mathrm{L2O}}(x;\, \zeta) := \eta_\theta\left(x - \tilde{W}^T(Ax - d)\right), \tag{30}$$

where $\zeta = \{\theta, \tilde{W}\}$ paramterizes the update operator with non-negative scalar θ and a matrix weight $W \in \mathbb{R}^{256\times 512}$. The tunable parameters $\Theta = (\theta^k, \hat{W}^k)_{k=1}^K$ include K scalars and K matrices. We take $K = 20$, i.e. we train a 20-layer L2O LISTA-CP model. To train the L2O LISTA-CP model, we use (16) as loss function with A mentioned above and $\tau = 0.01$.

Supplement to Subsection 5.4. We take $m = 500$, $n = 250$, $a_{ij} \sim \mathcal{N}(0,1)$. We use noise $\varepsilon \sim \mathcal{N}(0, 1/m)$. Each $d \sim \mathcal{D}_s$ used for training the neural network is sampled using $d = Ax^\star + \varepsilon$ with $x \sim \max(\mathcal{N}(0,1), 0)$. For unseen data $d \sim \mathcal{D}_u$ we sample $x^\star \sim \max(\mathcal{N}(5,5), 0)$. We sample 10,000 training samples from $\mathcal{D}_s$ to train the neural network and 1,000 samples from $\mathcal{D}_s$ and $\mathcal{D}_u$, respectively, for testing.

A.2. Proofs

This section contains a proof to the main theorem, subsequent corollary, and also a corollary regarding the safeguarding condition used for Linearized ADMM. We restate Theorem 3.1 in the Hilbert space setting. Then a proof is provided for it. We note again that this can be easily extended to the general Hilbert space setting (with weak convergence) and that Assumption 1 Part 3 can be weakened to assume that the limit infiumum of $\|x - T(x)\|$ is positive as $\|x\| \to \infty$.

Theorem 3.1. If $\{x^k\}$ is a sequence generated by the Safe-L2O method and Assumptions 1 and 2 hold, then

$$\lim_{k\to\infty} d_{\mathrm{Fix}(T)}\left(x^k\right) = 0, \tag{31}$$

that is, the sequence approaches the solution set. And, if $\{x^k\}$ contains a single cluster point, then $\{x^k\}$ converges to a point $x^\star \in \mathrm{Fix}(T)$ that is a solution to (1).

Proof. The proof is broken into two cases as follows.

Case 1: If the inequality in Line 9 holds finitely many times, then there exists an index beyond which the conventional method on Line 12 is always used to update x^k. In this case, for large k the Safe-L2O method takes the form of the classic KM Method, which is known to converge (e.g., see (Cegielski, 2012, Theorem 3.7.1), (Groetsch, 1972, Corollary 3), and (Bauschke & Combettes, 2017, Theorem 5.15)).

Case 2: Consider the case where the inequality in Line 9 holds infinitely many times. We proceed by first showing the sequence $\{x^k\}$ is bounded (Step 1). This is used to

[3] The code for LADMM experiment is based on public repo of (Xie et al., 2019), found at GitHub.

[4] We use the dictionary provided in the source code of (Xie et al., 2019), found at GitHub.

prove $\{x^k\}$ has a cluster point in $\text{Fix}(T)$ (Step 2). Results from these steps are then applied to obtain the desired limit (31) (Step 3).

Step 1: By Assumption 1 Part 3, there exists $R \in (0,\infty)$ sufficiently large to ensure

$$\|x\| > R \quad \Longrightarrow \quad \|x - T(x)\| > 1, \quad \text{for all } x \in \mathbb{R}^n. \tag{32}$$

Equivalently, we may write

$$\|x - T(x)\| \leq 1 \quad \Longrightarrow \quad \|x\| \leq R, \quad \text{for all } x \in \mathbb{R}^n. \tag{33}$$

By Assumption 2, there also exists $N_1 \in \mathbb{N}$ such that

$$\mu_k \leq 1, \quad \text{for all } k \geq N_1. \tag{34}$$

Fix any $z \in \text{Fix}(T)$. We claim

$$\|x^k - z\| \leq \max_{\ell \in [N_1]} \{2R,\ \|x^\ell - z\|\}, \quad \text{for all } k \in \mathbb{N}. \tag{35}$$

The result (35) holds trivially for all $k \in [N_1]$. Proceeding by induction, suppose (35) holds for some $k \geq N_1$. If the inequality in Line 9 holds, then (33), (34) and the update formula in Line 10 together imply $\|x^{k+1}\| \leq R$. Since $\|z - T(z)\| = 0$, (33) also implies $\|z\| \leq R$. Thus,

$$\begin{aligned} \|x^{k+1} - z\| &\leq \|x^{k+1}\| + \|z\| &(36)\\ &\leq 2R &(37)\\ &\leq \max_{\ell \in [N_1]} \{2R,\ \|x^\ell - z\|\}. &(38) \end{aligned}$$

If instead the update in Line 12 is applied, the averagedness of T implies there is $\tau \in (0,1)$ such that

$$\|x^{k+1} - z\|^2 \leq \|x^k - z\|^2 - \frac{1-\tau}{\tau}\|x^k - T(x^k)\|^2 \tag{39}$$

(e.g., see Prop. 4.35 in (Bauschke & Combettes, 2017) or Cor. 2.2.15 and Cor. 2.2.17 in (Cegielski, 2012)), and so

$$\|x^{k+1} - z\| \leq \|x^k - z\| \leq \max_{\ell \in [N_1]} \{2R,\ \|x^\ell - z\|\}. \tag{40}$$

Equations (38) and (40) together close the induction, from which (35) follows. Whence, for all $k \in \mathbb{N}$,

$$\|x^k\| \leq \|x^k - z\| + \|z\| \leq \max_{\ell \in [N_1]} \{2R,\ \|x^\ell - z\|\} + R, \tag{41}$$

which verifies the sequence $\{x^k\}$ is bounded.

Step 2: Because the inequality in Line 9 holds infinitely many times, there exists a subsequence $\{x^{q_k}\} \subseteq \{x^k\}$ satisfying

$$0 \leq \lim_{k\to\infty} \|x^{q_k} - T(x^{q_k})\| \leq \lim_{k\to\infty} \alpha\mu_k = 0, \tag{42}$$

from which the squeeze theorem asserts $\|x^{q_k} - T(x^{q_k})\| \to 0$. Since $\{x^k\}$ is bounded, so also is $\{x^{q_k}\}$. Thus, there exists a subsequence $\{x^{\ell_k}\} \subseteq \{x^{q_k}\}$ converging to a limit $p \in \mathbb{R}^n$. Then applying the fact $\text{Id} - T$ is 2-Lipschitz and $\|\cdot\|$ is continuous yields

$$\begin{aligned} 0 &= \lim_{k\to\infty} \|x^{\ell_k} - T(x^{\ell_k})\| &(43)\\ &= \left\| \lim_{k\to\infty} x^{\ell_k} - T\left(\lim_{k\to\infty} x^{\ell_k}\right)\right\| &(44)\\ &= \|p - T(p)\|, &(45) \end{aligned}$$

which implies $p \in \text{Fix}(T)$. That is, $\{x^k\}$ contains a cluster point $p \in \text{Fix}(T)$.

Step 3: Let $\varepsilon \in (0,1)$ be given. Following (Bauschke et al., 2014, Def. 2), define the ε-enlargement

$$\text{Fix}(T)_{[\varepsilon]} := \{x \in \mathbb{R}^n : d_{\text{Fix}(T)}(x) \leq \varepsilon\}. \tag{46}$$

Note $\text{Fix}(T)_{[\varepsilon]}$ is a nonempty closed and bounded subset of $\mathbb{R}^n$. This implies there exists $\tilde{R} \geq R$ such that

$$\text{Fix}(T)_{[\varepsilon/2]} \subset B(0, \tilde{R}), \tag{47}$$

where $B(0,\tilde{R})$ is the closed ball of radius $\tilde{R}$ centered at the origin. Set

$$\Omega := \left(B(0,\tilde{R}) - \text{Fix}(T)_{[\varepsilon/2]}\right) \cup \partial\text{Fix}(T)_{[\varepsilon/2]}, \tag{48}$$

Because $\mathbb{R}^n$ is finite dimensional and Ω is closed and bounded, Ω is compact. Thus, every continuous function obtains its infimum over Ω. In particular, we may set

$$\zeta = \min_{x\in\Omega} \|x - T(x)\|. \tag{49}$$

Note $\zeta > 0$ since $\Omega \cap \text{Fix}(T) = \emptyset$. Consequently, letting $\tilde{\zeta} := \min\{1, \zeta/2\}$ yields

$$\begin{aligned} \|x - T(x)\| \leq \tilde{\zeta} \quad &\Longrightarrow \quad x \in \text{Fix}(T)_{[\varepsilon]} \\ &\Longrightarrow \quad d_{\text{Fix}(T)}(x) \leq \varepsilon \end{aligned} \tag{50}$$

where the first implication holds because $x \in B(0,\tilde{R})$ by (33) and $x \notin \Omega$ by (49). By Assumption 2, there exists $N_2 \in \mathbb{N}$ such that

$$\mu_k \leq \tilde{\zeta}, \quad \text{for all } k \geq N_2. \tag{51}$$

By the result of Step 2, there exists $N_3 \geq N_2$ such that

$$\|x^{N_3} - p\| \leq \varepsilon \quad \Longrightarrow \quad d_{\text{Fix}(T)}(x^{N_3}) \leq \varepsilon. \tag{52}$$

We claim

$$d_{\text{Fix}(T)}(x^k) \leq \varepsilon, \quad \text{for all } k \geq N_3, \tag{53}$$

which, by the arbitrariness of ε, implies (31) holds. Indeed, inductively suppose (53) holds for some $k \geq N_3$. If the inequality in Line 9 holds, then (51) implies

$$\|x^{k+1} - T(x^{k+1})\| \leq \alpha\mu_k \leq \tilde{\zeta}, \tag{54}$$

and, by (50),

$$d_{\text{Fix}(T)}(x^{k+1}) \leq \varepsilon. \tag{55}$$

If instead the inequality in Line 9 does not hold, letting $P : \mathbb{R}^n \to \mathbb{R}^n$ be the projection operator onto $\text{Fix}(T)$ reveals

$$\begin{aligned} \|x^{k+1} - P(x^{k+1})\| &\leq \|x^{k+1} - P(x^k)\| \\ &\leq \|x^k - P(x^k)\| \\ &= d_{\text{Fix}(T)}(x^k) \\ &\leq \varepsilon, \end{aligned} \tag{56}$$

where the first inequality holds since the projection operator is defined as the point in $\text{Fix}(T)$ with minimum distance to x^{k+1} and the second inequality follows from (39), taking $T = P$ and $z = P(x^k)$. Note the left hand side of (56) equals the distance from x^{k+1} to $\text{Fix}(T)$. Therefore, (55) and (56) close the induction in each case, and (31) holds.

The limit (31) can be used in a similar manner to the work in Step 2 above to prove each cluster point of $\{x^k\}$ is in $\text{Fix}(T)$. Thus, if $\{x^k\}$ admits a unique cluster point, then the entire sequence converges to a point $x^\star \in \text{Fix}(T)$. □

We restate Corollary 3.1 below and then provide a proof.

Corollary 3.1. If $\{x^k\}$ is a sequence generated by the Safe-L2O method and Assumption 1 holds and $\{\mu_k\}$ is generated using a scheme outlined in Table 1, then Assumption 2 holds and, by Theorem 3.1, the limit (31) holds.

Proof. The proof is parsed into four parts, one for each particular choice of the sequence $\{\mu_k\}$ in Table 1, where we note "Recent Term" is a special case of "Recent Max" obtained by taking $m = 1$. Each proof part is completely independent of the others and is separated by italic text. However, to avoid excessive writing, in each section let $\Gamma \subseteq \mathbb{N}$ be the set of all indices for which the inequality in the conditional definitions of $\{\mu_k\}$ hold, the sequence $\{t_k\}$ be an ascending enumeration of Γ, and m_k be the number of times the inequality in the conditional definition of $\{\mu_k\}$ has been satisfied by iteration k.

Geometric Seqeunce. Define the sequence $\{\mu_k\}$ using, for each $k \in \mathbb{N}$, the Geometric Sequence update formula in Table 1. This implies

$$\mu_k = \alpha^{m_k}\mu_1. \tag{57}$$

Since Γ is infinite, $\lim_{k\to\infty} m_k = \infty$, and it follows that

$$\lim_{k\to\infty} \mu_k = \lim_{k\to\infty} (1-\delta)^{m_k}\mu_1 = 0 \cdot \mu_1 = 0, \tag{58}$$

i.e., Assumption 2 holds.

Arithmetic Average. Define the sequence $\{\mu_k\}$ using the AA update formula in Table 1. Then observe that, at each index t_k,

$$\begin{aligned} 0 \leq \mu_{t_k+1} &\leq \frac{\alpha\mu_{t_k} + m_{t_k}\mu_{t_k}}{m_{t_k}+1} \\ &= \left(1 - \frac{1-\alpha}{m_{t_k}+1}\right)\mu_{t_k} \\ &\leq \mu_{t_k}, \quad \text{for all } k \in \mathbb{N}. \end{aligned} \tag{59}$$

Since $\mu_{k+1} = \mu_k$ whenever $k \notin \Gamma$, (59) shows $\{\mu_k\}$ is monotonically decreasing. Consequently, using induction reveals

$$\begin{aligned} 0 &\leq \mu_{t_k} - \frac{1-\alpha}{m_{t_k}+1}\mu_{t_k} \\ &\leq \mu_1 - \sum_{\ell=1}^{k} \frac{(1-\alpha)\mu_{t_\ell}}{m_{t_\ell}+1} \\ &= \mu_1 - \sum_{\ell=1}^{k} \frac{(1-\alpha)\mu_{t_\ell}}{\ell+1} \quad \text{for all } k \in \mathbb{N}, \end{aligned} \tag{60}$$

where we note $m_{t_\ell} = \ell$ in the sum since m_ℓ increments once each time a modification occurs in the sequence $\{\mu_k\}$. By way of contradiction, suppose there exists $\tau \in (0, \infty)$ such that

$$\liminf_{k\to\infty} \mu_k \geq \tau > 0. \tag{61}$$

With the monotonicity of $\{\mu_k\}$, (60) implies

$$\sum_{\ell=1}^{k} \frac{(1-\alpha)\tau}{\ell+1} \leq \sum_{\ell=1}^{k} \frac{(1-\alpha)\mu_{t_\ell}}{\ell+1} \leq \mu_1, \quad \text{for all } k \in \mathbb{N}. \tag{62}$$

However, the sum on the left hand side becomes a divergent harmonic series as $k \to \infty$, contradicting the finite upper bound on the right hand side. This contradiction proves assumption (61) is false, from which it follows that

$$\liminf_{k\to\infty} \mu_k = 0. \tag{63}$$

By the monotone convergence theorem and nonnegativity of each μ_k, we deduce $\mu_k \to 0$, i.e., Assumption 2 holds.

Exponential Moving Average. Given $\theta \in (0,1)$, define the sequence $\{\mu_k\}$ using the EMA(θ) formula in Table 1. For each k when μ_{t_k} changes value, observe

$$\begin{aligned} \mu_{t_k+1} &= \theta\|x^{t_k+1} - T(x^{t_k+1})\| + (1-\theta)\mu_{t_k} \\ &\leq \theta\alpha\mu_{t_k} + (1-\theta)\mu_{t_k} \\ &= \theta(1-\alpha)\mu_{t_k}. \end{aligned} \tag{64}$$

This implies the sequence $\{\mu_k\}$ is nonincreasing and, when a decrease does occur, it is by a geometric factor of the current iterate. Through induction, it follows that

$$\mu_k \leq [\theta(1-\alpha)]^{m_k}\mu_1, \quad \text{for all } k \in \mathbb{N}. \tag{65}$$

Since Γ is infinite, $\lim\limits_{k\to\infty} m_k = \infty$. This, combined with the fact $\theta(1-\alpha) \in (0,1)$, implies

$$0 \leq \lim_{k\to\infty} \mu_k \leq \lim_{k\to\infty} [\theta(1-\alpha)]^{m_k}\mu_1 = 0 \cdot \mu_1 = 0, \tag{66}$$

from which Assumption 2 holds by the squeeze theorem.

Recent Max. Let $m \in \mathbb{N}$. Set Ξ_k to be the set of the most recent m indices in Γ, counting backwards from k, where $\{\mu_k\}$ is defined by the update formula in Table 1. When there are less than m indices in $\Gamma \cap \{1, 2, \ldots, k\}$, we let Ξ_k be all of the indices in the intersection. The sequence $\{\mu_k\}$ is monotonically decreasing since, for each k in Γ, the new term $\|x^k - T(x^k)\|$ is introduced so that $\|x^k - T(x^k)\| \in \Xi_{k+1}$, and this new term is no larger than the largest term in Ξ_k. All that remains is to show this sequence converges to zero. By way of contradiction, suppose there exists $\tau \in (0,\infty)$ such that

$$\liminf_{k\to\infty} \mu_k = \tau > 0. \tag{67}$$

Then choose

$$\varepsilon = \frac{(1-\alpha)\tau}{2\alpha}, \tag{68}$$

which implies

$$\alpha(\tau + \varepsilon) < \tau. \tag{69}$$

By (67) and the fact Γ is infinite, there exists $\tilde{N} \in \mathbb{N}$ with $\tilde{N} > m$ such that

$$\|\mu_{t_{\tilde{N}}} - \tau| < \varepsilon \quad \Longrightarrow \quad \mu_{t_{\tilde{N}}} < \tau + \varepsilon. \tag{70}$$

Then note each new element to Ξ_k is no larger than $\alpha\mu_{t_{\tilde{N}}}$. And, for any k after m such replacements occur,

$$\mu_k = \max_{\ell\in\Xi_k} \|S(x^\ell)\| \leq \alpha\mu_{t_{\tilde{N}}} \leq \alpha(\tau+\varepsilon) < \tau, \tag{71}$$

a contradiction to (67). This contradiction shows our assumption (67) must be false, and so

$$\liminf_{k\to\infty} \mu_k = 0. \tag{72}$$

By the monotone convergence theorem, we conclude Assumption 2 holds. □

Below is a lemma used to identify the safeguarding procedure for the LADMM method.

Lemma 1. Let $\{(x^k, z^k, \nu^k)\}$ be a sequence generated by LiADMM as in (25). If $\alpha\gamma\|B^tB\|_2 < 1$ and $\alpha\beta\|A^tA\|_2 < 1$, then for each index k there is an associated iterate ν^k such that the update ν^{k+1} is generated by applying an averaged operator T to ν^k with respect to the Euclidean norm, i.e., $\nu^{k+1} = T(\nu^k)$. In addition, for

$$\begin{aligned} P &:= \left(\frac{1}{\alpha\beta}\mathrm{Id} - A^TA\right)^{1/2}, \\ Q &:= \left(\frac{1}{\alpha\gamma}\mathrm{Id} - B^TB\right)^{1/2}, \end{aligned} \tag{73}$$

it holds

$$\|\nu^k - T(\nu^k)\| = \left\| \begin{bmatrix} Ax^{k+2} + Bz^{k+1} - d \\ P(x^{k+2} - x^{k+1}) \\ -Q(z^{k+1} - z^k) \end{bmatrix} \right\|. \tag{74}$$

Proof. We outline the proof as follows. We first derive a trio of updates that forms the application of an averaged operator for the ADMM problem (24) (Step 1). Next we rewrite the updates in a more meaningful manner using minimizations with f and g (Step 2). This formulation is then applied to a proximal ADMM problem (a special case of ADMM) that introduces auxiliary variables. This yields an explicit formula for (74) (Step 3). The remaining step uses substitution to transform the proximal ADMM formulation into the linearized ADMM updates in (25) (Step 4).

Step 1: The classic ADMM method applied to the problem (24) is equivalent to applying Douglas Rachford Splitting (DRS) to the problem

$$\min_\nu \underbrace{(A \rhd f)(\nu)}_{=:\tilde{f}(\nu)} + \underbrace{(B \rhd g)(d-\nu)}_{=:\tilde{g}(\nu)} = \min_\nu \tilde{f}(\nu) + \tilde{g}(\nu), \tag{75}$$

where $(A \rhd f)$ is the infimal postcomposition (e.g., see (Bauschke & Combettes, 2017))

$$(A \rhd f)(\nu) := \inf_{x\in\{\xi : A\xi = \nu\}} f(x). \tag{76}$$

This yields the iteration

$$\nu^{k+1} = \underbrace{\frac{1}{2}\left(\mathrm{Id} + R_{\alpha\partial\tilde{f}}R_{\alpha\partial\tilde{g}}\right)}_{=:T}(\nu^k) = T(\nu^k), \tag{77}$$

which may be rewritten in parts by

$$\begin{aligned} \zeta^{k+1/2} &= \mathrm{prox}_{\alpha^{-1}\tilde{g}}(\nu^k), \\ \zeta^{k+1} &= \mathrm{prox}_{\alpha^{-1}\tilde{f}}(2\zeta^{k+1/2} - \nu^k), \\ \nu^{k+1} &= \nu^k + \zeta^{k+1} - \zeta^{k+1/2}. \end{aligned} \tag{78}$$

This formulation reveals

$$\|\nu^k - T(\nu^k)\| = \|\zeta^{k+1} - \zeta^{k+1/2}\|. \tag{79}$$

Below we transform this expression into something meaningful.

Step 2: It can be shown that if the range of A^T intersected with the domain of the dual f^* is nonempty (i.e., $\mathcal{R}(A^t) \cap \text{ri dom}(f^\star) \neq \emptyset$), then

$$\zeta = \text{prox}_{A \triangleright f}(\nu) \tag{80}$$

if and only if

$$\zeta = Ax \;\text{ and }\; x \in \arg\min_{\xi} f(\xi) + \frac{1}{2}\|A\xi - \nu\|^2. \tag{81}$$

Also, for a function $B(x) = A(t-x)$ we have

$$\text{prox}_{\alpha B}(u) = t - \text{prox}_{\alpha A}(d - u). \tag{82}$$

These equivalences imply (78) may be rewritten as

$$\begin{aligned}
z^{k+1} &\in \arg\min_{z} g(z) + \frac{\alpha}{2}\|Bz - (d - \nu^k)\|^2, \\
\zeta^{k+1/2} &= d - Bz^{k+1}, \\
x^{k+2} &\in \arg\min_{x} f(x) + \frac{\alpha}{2}\|Ax - (2\zeta^{k+1/2} - \nu^k)\|^2, \\
\zeta^{k+1} &= Ax^{k+2}, \\
\nu^{k+1} &= \nu^k + \zeta^{k+1/2} - \zeta^{k+1}.
\end{aligned} \tag{83}$$

All instances of ζ^k and ζ^{k+1} may be removed upon substitution, i.e.,

$$\begin{aligned}
z^{k+1} &\in \arg\min_{z} g(z) + \frac{\alpha}{2}\|\nu^k + Bz - d\|^2, \\
x^{k+2} &\in \arg\min_{x} f(x) + \frac{\alpha}{2}\|\nu^k + Ax + 2(Bz^{k+1} - d)\|^2, \\
\nu^{k+1} &= \nu^k + (Ax^{k+2} + Bz^{k+1} - d).
\end{aligned} \tag{84}$$

Step 3: We proceed by rewriting the ADMM problem with the introduction of the auxiliary matrices

First note that our hypothesis implies the inverses of P and Q are defined and the square root can be taken since the matrices are symmetric. In addition, P and Q are positive definite. The ADMM problem (24) is equivalent to

$$\min_{x,z} f(x) + g(z) \tag{85}$$

subject to the constraint

$$\begin{bmatrix} A & 0 \\ P & 0 \\ 0 & \text{Id} \end{bmatrix} \begin{bmatrix} x \\ \tilde{x} \end{bmatrix} + \begin{bmatrix} B & 0 \\ 0 & \text{Id} \\ Q & 0 \end{bmatrix} \begin{bmatrix} z \\ \tilde{z} \end{bmatrix} = \begin{bmatrix} d \\ 0 \\ 0 \end{bmatrix}. \tag{86}$$

Using the same update formula as in Step 2 yields

$$\begin{aligned}
(z^{k+1}, \tilde{z}^{k+1}) &\in \arg\min_{(z,\tilde{z})} g(z) + \frac{\alpha}{2}\|\nu_1^k + Bz - d\|^2 \\
&\quad + \frac{\alpha}{2}\|\nu_2^k + \tilde{z}\|^2 + \frac{\alpha}{2}\|\nu_3^k + Qz\|^2, \\
(x^{k+2}, \tilde{x}^{k+2}) &\in \arg\min_{(x,\tilde{x})} f(x) \\
&\quad + \frac{\alpha}{2}\|\nu_1^k + Ax + 2(Bz^{k+1} - d)\|^2 \\
&\quad + \frac{\alpha}{2}\|\nu_2^k + Px + 2\tilde{z}^{k+1}\|^2 \\
&\quad + \frac{\alpha}{2}\|\nu_3^k + \tilde{x} + 2Qz^{k+1}\|^2, \\
\nu_1^{k+1} &= \nu_1^k + (Ax^{k+2} + Bz^{k+1} - d), \\
\nu_2^{k+1} &= \nu_2^k + (P\tilde{x}^{k+2} + \tilde{z}^{k+1}), \\
\nu_3^{k+1} &= \nu_3^k + (\tilde{x}^{k+2} + Qz^{k+1}),
\end{aligned} \tag{87}$$

where $\nu^k = (\nu_1^k, \nu_2^k, \nu_3^k)$. Simplifying reveals

$$\begin{aligned}
z^{k+1} &\in \arg\min_{z} g(z) + \frac{\alpha}{2}\|\nu_1^k + Bz - d\|^2 \\
&\quad + \frac{\alpha}{2}\|\nu_3^k + Qz\|^2, \\
\tilde{z}^{k+1} &= -\nu_2^k, \\
x^{k+2} &\in \arg\min_{(x,\tilde{x})} f(x) + \frac{\alpha}{2}\|\nu_2^k + Px + 2\tilde{z}^{k+1}\|^2 \\
&\quad + \frac{\alpha}{2}\|\nu_1^k + Ax + 2(Bz^{k+1} - d)\|^2 \\
\tilde{x}^{k+2} &= -\nu_3^k - 2Qz^{k+1}, \\
\nu_1^{k+1} &= \nu_1^k + (Ax^{k+2} + Bz^{k+1} - d), \\
\nu_2^{k+1} &= \nu_2^k + (Px^{k+2} + \tilde{z}^{k+1}) = Px^{k+2}, \\
\nu_3^{k+1} &= \nu_3^k + (\tilde{x}^{k+2} + Qz^{k+1}) = -Qz^{k+1}.
\end{aligned} \tag{88}$$

Simplifying once more gives

$$\begin{aligned}
z^{k+1} &\in \arg\min_{z} g(z) + \frac{\alpha}{2}\|\nu_1^k + Bz - d\|^2 \\
&\quad + \frac{\alpha}{2}\|Q(z - z^k)\|^2, \\
x^{k+2} &\in \arg\min_{(x,\tilde{x})} f(x) + \frac{\alpha}{2}\|P(x - x^{k+1})\|^2 \\
&\quad + \frac{\alpha}{2}\|\nu_1^k + Ax + 2(Bz^{k+1} - d)\|^2 \\
\nu_1^{k+1} &= \nu_1^k + (Ax^{k+2} + Bz^{k+1} - d), \\
\nu_2^{k+1} &= Px^{k+2}, \\
\nu_3^{k+1} &= -Qz^{k+1}.
\end{aligned} \tag{89}$$

Thus,

$$\|\nu^{k+1} - \nu^k\| = \left\| \begin{bmatrix} Ax^{k+2} + Bz^{k+1} - d \\ P(x^{k+2} - x^{k+1}) \\ -Q(z^{k+1} - z^k) \end{bmatrix} \right\|. \tag{90}$$

Step 4: We now derive the form of the linearized ADMM updates. Let $u^k = \alpha(\nu_1^k - Ax^{k+1})$. Then

$$u^{k+1} = u^k + \alpha(Ax^{k+1} + Bz^{k+1} - d) \tag{91}$$

and

$$\begin{aligned} z^{k+1} &\in \arg\min_z g(z) + \frac{\alpha}{2}\|Q(z - z^k)\|^2 \\ &+ \frac{\alpha}{2}\|\alpha^{-1}u^k + Ax^{k+1} + Bz - d\|^2 \\ &= \arg\min_z g(z) + \frac{\alpha}{2}\left\langle z - z^k, Q^2(z - z^k)\right\rangle + \langle Bz, u^k\rangle \\ &+ \frac{\alpha}{2}\|Ax^{k+1} + Bz - d\|^2 \\ &= \arg\min_z g(z) + \langle Bz, u^k + \alpha(Ax^{k+1} + Bz^k - d)\rangle \\ &+ \frac{1}{2\gamma}\|z - z^k\|^2 \\ &= \text{prox}_{\gamma g}\left(z^k - \gamma B^T(u_1^k + \alpha(Ax^{k+1} + Bz^k - d)\right). \end{aligned} \tag{92}$$

In similar fashion, we deduce

$$\begin{aligned} x^{k+2} &\in \arg\min_x f(x) + \frac{\alpha}{2}\|P(x - x^{k+1})\|^2 \\ &+ \frac{\alpha}{2}\|\nu_1^k + Ax + 2(Bz^{k+1} - d)\|^2 \\ &= \arg\min_x f(x) + \frac{\alpha}{2}\left\langle x - x^{k+1}, P^2(x - x^{k+1}\right\rangle \\ &+ \frac{\alpha}{2}\|\alpha^{-1}u_1^k + Ax^{k+1} + Ax + 2(Bz^{k+1} - d)\|^2 \\ &= \arg\min_x f(x) + \frac{1}{2\beta}\|x - x^{k+1}\|^2 - \frac{\alpha}{2}\|Ax\|^2 \\ &+ \langle Ax, \alpha Ax^{k+1}\rangle + \frac{\alpha}{2}\|\alpha^{-1}u^k + Ax + Bz^{k+1} - d\|^2 \\ &= \text{prox}_{\beta f}\left(x^{k+1} - \beta A^T(u^k + \alpha(Ax^{k+} + Bz^{k+1} - d)\right). \end{aligned} \tag{93}$$

Upon reordering the updates in (91), (92) and (93) to obtain the appropriate dependencies, we obtain (25), as desired.

□