Safeguarded Learned Convex Optimization

Howard Heaton[*1], **Xiaohan Chen**[*2], **Zhangyang Wang**[2], **Wotao Yin**[3]
[1]Typal Research, Typal LLC
[2]Department of Electrical and Computer and Engineering, The University of Texas at Austin
[3]Alibaba US, DAMO Academy, Decision Intelligence Lab

Abstract

Applications abound in which optimization problems must be repeatedly solved, each time with new (but similar) data. Analytic optimization algorithms can be hand-designed to provably solve these problems in an iterative fashion. On one hand, data-driven algorithms can "learn to optimize" (L2O) with much fewer iterations and similar cost per iteration as general-purpose optimization algorithms. On the other hand, unfortunately, many L2O algorithms lack converge guarantees. To fuse the advantages of these approaches, we present a Safe-L2O framework. Safe-L2O updates incorporate 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 is activated only when the data-driven L2O updates would perform poorly or appear to diverge. This yields the numerical benefits of employing machine learning to create rapid L2O algorithms while still guaranteeing convergence. Our numerical examples show convergence of Safe-L2O algorithms, even when the provided data is *not* from the distribution of training data.

Introduction

Solving scientific computing problems often requires use of efficient optimization algorithms. Data-driven algorithms can execute in far fewer iterations and with similar cost per iteration as state-of-the-art general purpose algorithms. Inspired by one such algorithm, ISTA, (Gregor and LeCun 2010) proposed treating entries in fixed matrices/vectors of the algorithm as learnable parameters that can vary by iteration. These entries were fine-tuned to obtain optimal performance on a data set for a fixed number of iterations. Empirically, this approach converged with a roughly a 20-fold reduction in computational cost compared to the original analytic algorithm. Related works also showed numerical success (reviewed below). These efforts open the door to a new class of algorithms and analyses. Analytic optimization results often provide worst-case convergence rates, and limited theory exists pertaining to instances drawn from a common distribution. Most L2O methods have little or no convergence guarantees, especially on data *distinct* from what is seen in training. Indeed, LISTA and ALISTA (reviewed below) have linear convergence to a sparse signal when training finds proper parameters, *i.e.* it is possible they converge. This work addresses the inquiry:

Can a safeguard be added to L2O algorithms to guarantee convergence without significantly hindering performance?

A safeguard is anything that identifies when a "bad" L2O update (due to poorly trained parameters, lack of generalization, out-of-distribution inputs, or any other reasons) would occur and what to do in place of that "bad" update. Informally, what a safeguard does can be summarized by generating a sequence $\{x^k\}$ with updates of the form

$$x^{k+1} = \begin{cases} \text{L2O Update} & \text{if update is ``good''} \\ \text{Fallback Update} & \text{otherwise.} \end{cases} \tag{1}$$

We provide an affirmative answer to the question for convex problems with gradient and/or proximal oracles by providing such a safeguard and replacing "bad" L2O updates with updates from analytic methods. Since a trade-off is formed between per iteration costs and ensuring convergence, below we clarify properties for a "practical" L2O safeguard.

1. The safeguard should ensure certain forms of *worst-case* convergence similar to analytic algorithms.
2. The safeguard must only use known quantities related to convex problems (*e.g.* objective values, gradient norms).
3. Both L2O and Safe-L2O schemes should perform identically on "good" data with comparable per-iteration costs.
4. The safeguard should kick in only when "bad" L2O updates would otherwise occur.

The core 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 (*i.e.* are not monotonically improving). Loosely speaking, this means there are cases 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 can be acceptable is we are interested in the final output of the L2O algorithm and L2O schemes may learn "shortcuts." From this insight, we deduce a safeguard should exhibit a form of trailing behavior, *i.e.* measure progress of previous iterates and

[*]These authors contributed equally.

only require that updates are "good" on average. If the safeguard triggers too often, then the Safe-L2O scheme's flexibility and performance are limited. If it triggers too rarely, then the Safe-L2O scheme may exhibit highly oscillatory behavior and converge slowly.

In addition to L2O updates, our method uses a safeguard condition with the update formula from a conventional algorithm. When the "good" 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 applied often 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 (see Figure 1).

Review of L2O Methods. A seminal L2O work in the context of sparse coding was by (Gregor and LeCun 2010). Numerous follow-up papers also demonstrated empirical success at constructing rapid regressors approximating iterative sparse solvers for compression, nonnegative matrix factorization, compressive sensing and other applications (Sprechmann, Bronstein, and Sapiro 2015; Wang, Ling, and Huang 2016; Wang et al. 2016; Hershey, Roux, and Weninger 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. Some works have interpreted LISTA in various ways to provide proofs of different convergence properties (Giryes et al. 2018; Moreau and Bruna 2017). Others have investigated structures related to LISTA (Xin et al. 2016; Blumensath and Davies 2009; Borgerding, Schniter, and Rangan 2017; Metzler, Mousavi, and Baraniuk 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. 2019), which further simplified the weight conditions 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. We refer readers to the recent survey (Chen et al. 2022) for a comprehensive L2O overview. Our work is about optimization, *i.e.* minimization of an objective function, while some L2O works focus on inverse problems wherein optimization is used as a surrogate.

Our safeguarding scheme is related to existing works in Krasnosel'skiĭ-Mann (KM) methods. The SuperMann scheme (Themelis and Patrinos 2019) presents a KM method that safeguards in a more hierarchical manner than ours and solely refers to the current iterate residuals (plus a summable sequence). Additionally, a similar safeguarding setup has been used for Anderson accelerated KM methods (Zhang, O'Donoghue, and Boyd 2018). These methods are not designed with L2O in mind and differ from our approach both in the assumptions used and update formulas.

Our Contribution. We provide a simple framework, Safe-L2O, for wrapping data-driven algorithms with 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 give multiple safeguarding procedures in a general setting and a simple procedure for utilizing machine learning methods to instill knowledge from available data.

Fixed Point Methods

Fixed-point iteration abstracts most of the convex optimization methods that are based on gradient and/or proximal oracles, and those methods are targets of recent L2O accelerations. Our method is based on examining the fixed-point residual, and so it applies to a wide variety of L2O methods, even complicated first-order methods (*e.g.* ADMM and primal-dual methods) where the underlying problems can include constraints. To provide a brief background, here we overview fixed point methods. Denote the set of fixed points of each operator $T\colon \mathbb{R}^n \to \mathbb{R}^n$ by $\text{fix}(T) \triangleq \{x : Tx = x\}$. For an operator T with a nonempty fixed point set (*i.e.* $\text{fix}(T) \neq \emptyset$), we consider the fixed point problem

$$\text{Find } x^\star \text{ such that } x^\star \in \text{fix}(T). \tag{2}$$

See Table 1 for examples of the operator T in convex minimization. We focus on fixed point iteration to give a general approach for creating sequences that converge to solutions of (2) and, thus, of the corresponding optimization problem.

The following definitions are used throughout. An operator $T\colon \mathbb{R}^n \to \mathbb{R}^n$ is *nonexpansive* if it is 1-Lipschitz,[1] *i.e.*

$$\|T(x) - T(y)\| \le \|x - y\|, \quad \text{for all } x, y \in \mathbb{R}^n. \tag{3}$$

An operator T is *averaged* if there exists $\vartheta \in (0,1)$ and a nonexpansive operator $Q : \mathbb{R}^n \to \mathbb{R}^n$ such that $T = (1-\vartheta)\mathrm{I} + \vartheta Q$, with I the identity. A classic theorem (see Theorem 1) states sequences generated by successively applying an averaged operator converge to a fixed point. In this work, each operator T is averaged. This method comes from (Krasnosel'skiĭ 1955) and (Mann 1953), which yielded adoption of the name *Krasnosel'skiĭ-Mann* (KM) method.

Theorem 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 $x^\star \in \text{fix}(T)$ to (2) such that the sequence $\{x^k\}$ converges to $x^\star$.

We briefly discuss the use of resolvents in conventional algorithms. Consider a convex function $f\colon \mathbb{R}^n \to \mathbb{R}$ with subgradient ∂f and $\alpha > 0$, the *resolvent* $J_{\alpha\partial f}$ of $\alpha\partial f$ is defined by $J_{\alpha\partial f}(x) \triangleq (\mathrm{I} + \alpha\partial f)^{-1}(x)$, *i.e.*

$$J_{\alpha\partial f}(x) = \{y : (x-y)/\alpha \in \partial f(y)\} \tag{5}$$

and the *reflected resolvent* of ∂f is

$$R_{\alpha\partial f}(x) \triangleq (2J_{\alpha\partial f} - \mathrm{Id})(x) = 2J_{\alpha\partial f}(x) - x. \tag{6}$$

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

$$J_{\alpha\partial f}(x) = \text{prox}_{\alpha f}(x) \triangleq \operatorname*{arg\,min}_{z\in\mathbb{R}^n} \alpha f(z) + \frac{1}{2}\|z-x\|^2. \tag{7}$$

[1]The Euclidean norm on $\mathbb{R}^n$ is denoted by $\|\cdot\|$.

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

Table 1: 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 () and (6), respectively. The block matrix M is $M = [\alpha^{-1}\mathrm{Id}, A^T; -A, \beta^{-1}\mathrm{Id}]$. In each case, $\mathcal{L}$ is the Lagrangian associated with the presented problem.

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 (Bauschke and Combettes 2017). Table 1 provides examples of these operators in well-known algorithms.

Safeguarded L2O Method

Algorithm 1 L2O Network (No Safeguard)

```
1: $\mathcal{L2O}(d;\ \Theta)$ :
2:     $x^1 \leftarrow \tilde{x}$                            ◁ Initialize inference
3:     for  $k = 1, 2, \ldots, K$                ◁ Loop for each layer
4:         $x^{k+1} \leftarrow T_{\Theta^k}(x^k; d)$           ◁ L2O Update
5:     return $x^{K+1}$                         ◁ Output inference
```

This section presents the Safe-L2O framework. The safeguard acts as a wrapper around a data-driven algorithm, which is formulated in practice as a neural network. Each L2O operator, denoted throughout by T_{Θ^k}, is parameterized by layerwise weights $\Theta = (\Theta^1, ..., \Theta^K)$. Input data d is used to define an optimization problem (*e.g.* the measurement vector in a least squares problem). To make this dependence clear, we often write $T(\cdot;\ d)$. Often $T_{\Theta^k}(\cdot; d)$ can be viewed as forming one or multiple layers of a feed forward network. Thus, we interpret $\mathcal{N}_\Theta(d)$ in Algorithm 1 as a feed forward network. In addition to an L2O operator T_{Θ^k}, our Safe-L2O method uses a fallback operator T (unrelated to T_{Θ^k}) 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 safeguarding schemes is related to a combination of $\|y^i - T(y^i; d)\|$ and $\|y^i - x^i\|$ among previous iterates $i = 1, \ldots, k$, where $y^i = T_{\Theta^i}(x^i; d)$.

Algorithm 2 Safeguarded L2O (Safe-L2O)

```
 1: Safe-L2O($\mathcal{L2O}(d;\Theta), T, \alpha,\ \beta$)
 2:     $x^0 \leftarrow \tilde{x}, \quad x^1 \leftarrow \tilde{x}, \quad y^1 \leftarrow T_{\Theta^1}(x^1), \quad k \leftarrow 1$
 3:     $\mu_1 \leftarrow \alpha^{-1} \cdot (\|y^1 - T(y^1; d)\| + \beta\|y^1 - x^1\|)$
 4:     while $\|x^k - x^{k-1}\| > \varepsilon$ or $k = 1$
 5:         $y^k \leftarrow T_{\Theta^k}(x^k; d)$
 6:         if $\|y^k - T(y^k; d)\| + \beta\|y^k - x^k\| \le \alpha\mu_k$
 7:             $x^{k+1} \leftarrow y^k$
 8:         else
 9:             $x^{k+1} \leftarrow T(x^k; d)$
10:         Update safeguard $\mu_{k+1}$
11:         $k \leftarrow k + 1$
12:     return $x^k$
```

We propose the Safe-L2O scheme in Algorithm 2. As shown in Line 1, a safeguarded L2O operator consists of an L2O network $\mathcal{L2O}(d;\Theta)$, a fallback operator T, and a parameter $\alpha \in (0,1)$. Here $\Theta = (\Theta^1, \ldots, \Theta^K)$ forms layerwise weights Θ^k that define the L2O update T_{Θ^k} at the k-th iteration. These weights are trained beforehand via standard training methods (see the Appendices). Table 1 shows some choices of the fallback operator T for different optimization problems and algorithms In Line 2, the initial iterate x^1 is chosen to be an arbitrary (but fixed) vector $\tilde{x}$. The initial iterate μ_1 of the safeguard sequence $\{\mu_k\}$ is initialized using the initial iterate x^1, an L2O update y^1, and the fallback operator T in Line 3. From Line 4 to Line 11, a repeated loop occurs to compute each update x^{k+1}. In Line 5 the L2O operator is applied to the current iterate x^k get a tentative update y^k. This y^k is "good" if the the inequality in Line 6 holds. In such a case, the L2O update is assigned to x^{k+1} in Line 7. Otherwise, the fallback T is used to get update x^{k+1}

in Line 9. The initial iterate μ_1 is defined so $x^2 = y^1$, *i.e.* the first L2O update is always accepted. The safeguard is updated in Line 10. Note α gives flexibility in how quickly an L2O update must converge to be "good" (smaller α requires faster convergence) and $\beta > 0$ ensures L2O residuals are summable. Table 2 gives schemes to update the safeguard.

Below are standard assumptions used to prove the main result. The first enables a fixed point formulation.

Assumption 1. *For input data d, the optimization problem has a solution and there is an operator T such that i) $\text{fix}(T(\cdot; d))$ is the solution set and ii) $T(\cdot; d)$ is averaged.*

The next assumption ensures used L2O updates approach solutions. This is done by computing the safeguard value, which is a fixed point residual with the fallback operator.

Assumption 2. *Here $\alpha \in [0, 1)$, $\beta \in (0, \infty)$ and the safeguard $\{\mu_k\}$ is monotonically decreasing such that*

$$\|T(x^k; d) - x^k\| \leq \mu_k, \quad \text{for all } k \in \mathbb{N}, \tag{8}$$

and there exists $\zeta \in (0, 1)$ such that

$$\mu_{k+1} \leq \zeta\mu_k, \quad \text{whenever } x^{k+1} \text{ is an L2O update.} \tag{9}$$

Our proposed methods for choosing the sequence $\{\mu_k\}$ satisfy Assumption 2 (see Table 2). These methods are *adaptive* in the sense that each μ_k depends upon the weights Θ^k, iterate x^k and (possibly) previous weights and iterates. Each safeguard parameter μ_k also remains constant in k except for when the sum of residual norms $\|x^{k+1} - T(x^{k+1})\|$ and $\|y^k - x^k\|$ 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 the residual norm to increase in k from time to time. This trailing behavior provides flexibility to the L2O updates. Our main result is below and is followed by a corollary justifying use of the schemes in Table 2 (both proven in the appendix).

Theorem 2. *If $\{x^k\}$ is a sequence generated by the inner loop in Safe-L2O and Assumptions 1 and 2 hold, then $\{x^k\}$ converges to a limit $x_d^\star \in \text{fix}(T(\cdot; d))$, i.e. $x^k \to x_d^\star$.*

Corollary 1. *If $\{x^k\}$ is generated by the inner loop in Safe-L2O and Assumption 1 holds, and $\{\mu_k\}$ is generated using a scheme outlined in Table 2 with $\alpha \in [0, 1)$ and $\beta \in (0, \infty)$, then $x^k \to x_d^\star \in \text{fix}(T(\cdot; d))$.*

We summarize the safeguard schemes in Table 2 as follows. The GS method decreases μ_k be a fixed geometric factor at each update. The EMA method exponentially averages all past and current residual sums where μ_k is/was modified. The RT method sets μ_k to be the last residual norm sum to be "good", *i.e.* satisfy the C_k inequality. We find EMA to be the most practical safeguard due to its adaptive nature.

Remark 1. *The appropriate frequency for the safeguard to trigger 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. To avoid possible confusions, note we are not trying to prove the convergence of any standalone L2O algorithm. We instead 1) alarm on an L2O update when it may break convergence, 2) replace it with a fallback update, and 3) show the resulting "hybrid optimization" converges to a solution of the provided optimization problem.*

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. Input d is usually the *observation* we have, based on which we optimize over the variable of interest. For example, the LASSO problem in (12) is used for sparse coding, where the goal is to recover a unknown sparse vector $x^\star$ from its noisy measurements $d = Ax^\star + \varepsilon$. Assuming A is known beforehand, the input to the Safe-L2O model is the observation d. In other cases, the dictionary A can change and also be part of the input to the model. We include case-by-case discussions about what the inputs for each numerical example.

Each layer of the Safe-L2O model 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. The set over which Θ is minimized, may be chosen with great flexibility. For each application of the algorithm, the fallback operator depends upon the data 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 $\phi(x^K; d)$ is minimized among $d \sim \mathcal{D}$, where $\phi(\cdot\,; d) : \mathbb{R}^n \to \mathbb{R}$ is an appropriate cost function and K is a fixed positive integer. Mathematically, this means $\Theta^\star$ solves the problem

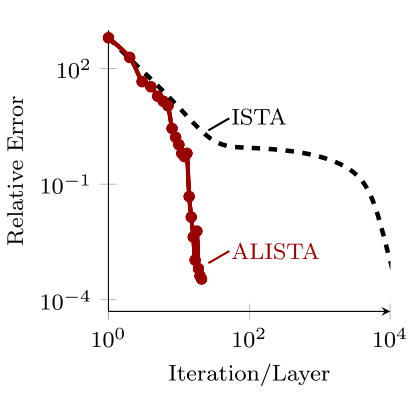
$$\min_{\Theta} \mathbb{E}_{d\sim\mathcal{D}}[\phi(x^K(\Theta;\ d);\ d)], \tag{10}$$

where we emphasize the dependence of x^K on Θ and d by writing $x^K = x^K(\Theta;\ d)$. Examples for ϕ include the original objective function (*i.e.* $\phi(x;\ d) = f(x;\ d)$) and the fixed point residual $\|x - T(x;\ d)\|$. We approximately solve (10) by sampling data $\{d^n\}_{n=1}^N$ from $\mathcal{D}$ and minimizing an empirical loss function. Summaries for training are outlined in the appendices. Note different learning problems than (10) may be used (*e.g.* the min-max problem used by adversarial networks (Goodfellow et al. 2014)).

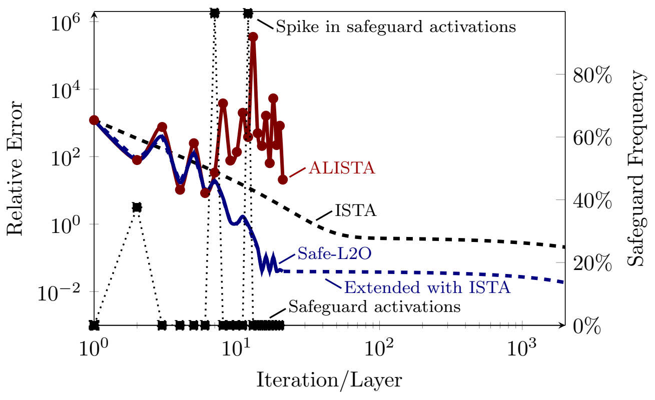
Numerical Examples

This section presents examples using Safe-L2O.[2] 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 L2O operators is not theoretically justified. We first use $\mathcal{L2O}$ from ALISTA (Liu et al. 2019) on a synthetic 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. In all three types of problems, the input data d to the L2O models is generated by $d = Ax^\star + \varepsilon$ where ε is white Gaussian noise and $x^\star$ is the hidden variable that we want to recover through a dictionary A, which is generated and beforehand and fixed. In these experiments, the ground-truth vector $x^\star$ are sampled from a distribution, which characterizes the distribution of data of

[2]Code is on GitHub: github.com/VITA-Group/Safe_L2O



(a) Performance on *seen* distribution



(b) Performance on *unseen* distribution

Figure 1: Plot of error versus iteration for ALISTA example. Here ISTA is the classic algorithm, ALISTA is the L2O operator in Algorithm 1 and Safe-L2O is the safeguarded version of ALISTA in Algorithm 2. 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 only when $k = 2$, $k = 7$, and $k = 12$.

NAME	UPDATE FORMULA
Geometric Sequence GS(θ)	$\mu_{k+1} = \begin{cases} \theta\mu_k & \text{if } C_k \text{ holds,} \\ \mu_k & \text{otherwise.} \end{cases}$ *Decrease μ_k by factor θ for "good" residuals.*
Recent Term RT	$\mu_{k+1} = \begin{cases} \|x^{k+1} - T(x^{k+1}; d)\| + \beta\|x^{k+1} - x^k\| & \text{if } C_k \text{ holds,} \\ \mu_k & \text{otherwise.} \end{cases}$ *Take μ_k to be most recent "good" residual.*
Exponential Moving Average EMA(θ)	$\mu_{k+1} = \begin{cases} \theta\left(\|x^{k+1} - T(x^{k+1}; d)\| + \beta\|x^{k+1} - x^k\|\right) + (1-\theta)\mu_{k-1} & \text{if } C_k \text{ holds,} \\ \mu_k & \text{otherwise.} \end{cases}$ *Exponentially average μ_k with the latest "good" residuals.*

Table 2: Rules for updating μ_k. Here $\alpha, \theta \in (0, 1)$, $\beta \in (0, \infty)$, and C_k is the statement $\|y^k - T(y^k;\ d)\| + \beta\|y^k - x^k\| \le \alpha\mu_k$.

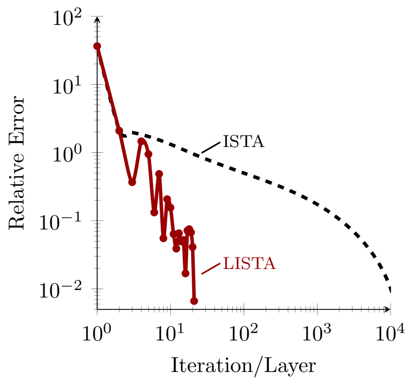
interest along with the dictionary. We denote the distribution of the input data as $\mathcal{D}$. Besides these "linear" examples, we also validate Safe-L2O on a distribution of LASSO problems where the dictionary A also changes and is part of the input to the L2O models. In this case, the distributions of the dictionaries and ground-truth vectors together characterize the input distribution $\mathcal{D}$.

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

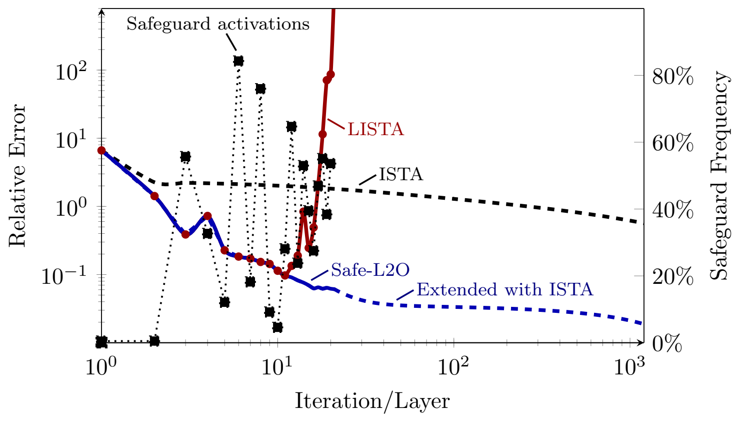
$$\text{Relative Error} = \mathcal{R}_{f,\mathcal{D}}(x) \triangleq \frac{\mathbb{E}_{d\sim\mathcal{D}}[f(x;\ d) - f_d^\star]}{\mathbb{E}_{d\sim\mathcal{D}}[f_d^\star]}, \quad (11)$$

where the expectations are estimated numerically (see the appendices for details). We use (11) rather than the expectation of relative error to avoid high sensitivity to outliers.

Our numerical results are shown 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 as compared to conventional optimization 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, with the reference axis on the right hand side of the plots. 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. As Safe-L2O convergence holds whenever $\beta > 0$, we can set β to be arbitrarily small (*e.g.* below machine precision); for simplicity, we use $\beta = 0$ in the experiments (as, even in this case, it can be shown that iterates approach the solution set). Implementation details for each experiment are in the appendices.



(a) Performance on *seen* distribution



(b) Performance on *unseen* distribution

Figure 2: Plot of error versus iteration for LISTA denoising. Here ISTA is the classic algorithm, LISTA is the L2O operator in Algorithm 1 and Safe-L2O is the safeguarded version of LISTA in Algorithm 2. 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 $k > 2$.

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 reasonably estimated by solving the LASSO problem

$$\min_{x\in\mathbb{R}^n} f(x;\, d) \triangleq \frac{1}{2}\|Ax - d\|_2^2 + \tau\|x\|_1, \tag{12}$$

where $\|\cdot\|_2$ and $\|\cdot\|_1$ are the ℓ_2 and ℓ_1 norms, respectively. A classic method for solving (12) is the iterative shrinkage thresholding algorithm (ISTA) (*e.g.* see (Daubechies, Defrise, and Mol 2004)).[3] Liu et al. (2019) present the L2O scheme ALISTA that we implement here. This L2O model $\mathcal{L2O}$ is parameterized by $\Theta^k = (\theta_k, \gamma_k) \in \mathbb{R}^2$.

Linearized ADMM

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

$$\min_{x\in\mathbb{R}^n} \|Ax - d\|_1 + \tau\|x\|_1. \tag{13}$$

The L2O scheme (D-LADMM) and fallback linearized ADMM (LiADMM) operator T are in the appendices along with implementation details. Plots are provided in Figure 3.

LISTA for Natural Image Denoising

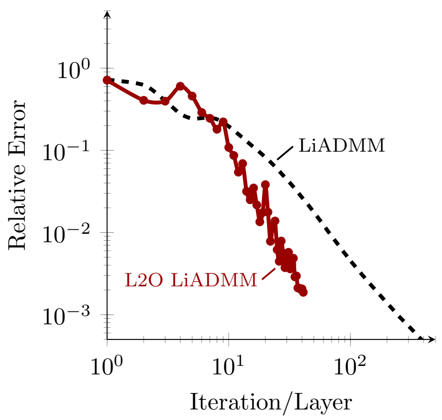
To evaluate our safeguarding mechanism in a more realistic setting, we apply safeguarded LISTA to a natural image denoising problem. In this subsection, we learn a LISTA-CP model (Chen et al. 2018) to perform natural image denoising. During training, L2O LISTA-CP model is trained to recover clean images from their Gaussian noisy counterparts by solving (12). In (12), 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 and 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.

AdaLISTA: Dictionary as Part of Inputs

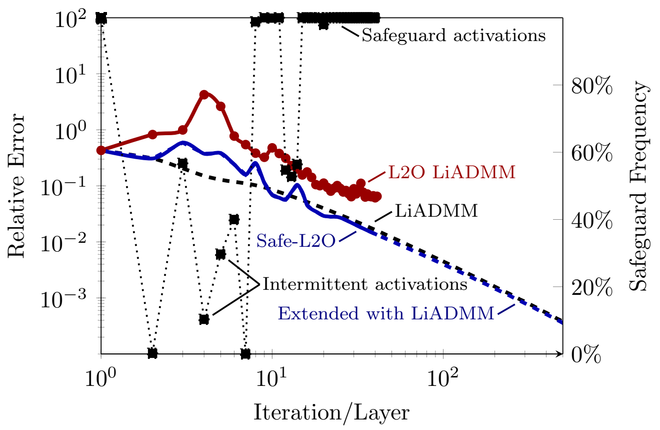
Here we consider the same LASSO problem (12) as in Subsection but make the dictionary A part of the inputs to the L2O model (*i.e.* able to change across samples). Aberdam, Golts, and Elad (2021) present a new L2O scheme AdaLISTA that is trained to quickly solve a distribution of LASSO problems with varying dictionaries. AdaLISTA has a different parameterization scheme from the original LISTA (Gregor and LeCun 2010) to enable the adaptivity to the dictionaries. The L2O model $\mathcal{L2O}$ in AdaLISTA is parameterized by $\zeta = (\theta, \gamma) \in \mathbb{R}^2$ and two weight matrices $W_1, W_2 \in \mathbb{R}^{m\times m}$ where the dictionary A has shape $m \times n$. The two matrices are shared by operators in all iterations.

We mainly follow the settings in (Aberdam, Golts, and Elad 2021). Specifically, we let $x^\star \in \mathbb{R}^{70}$ be sparse vectors with random supports of cardinality $s = 6$ and a single **fixed** dictionary $A' \in \mathbb{R}^{50\times 70}$. We assume access is given to noiseless linear measurements $d \in \mathbb{R}^{50} = Ax^\star$, where A is uniformly sampled from all column-permuted variants of A'. Figure 4 and 5 show summary plots, from which we have similar observations as in the ALISTA experiments with a single fixed dictionary.

[3]This is a special case of the proximal-gradient in Table 1.

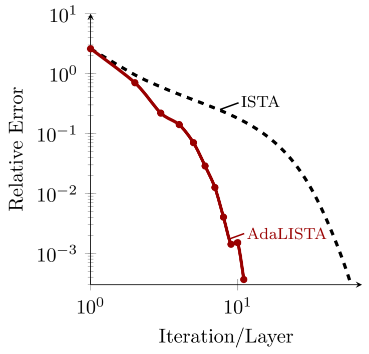


(a) Performance on *seen* distribution



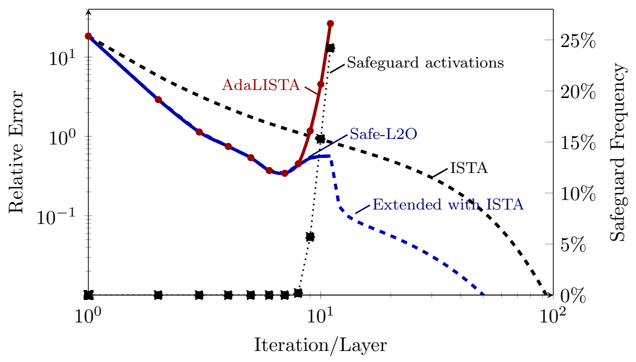
(b) Performance on *unseen* distribution

Figure 3: Plot of error versus iteration for LiADMM. Here LiADMM is the classic algorithm, L2O LiADMM is the L2O operator in Algorithm 1 and Safe-L2O is the safeguarded version of L2O LiADMM in Algorithm 2. 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. The safeguard is used about 10% and 30% of the time when $k = 4$ and $k = 5$, respectively.



(a) Performance on *seen* distribution

Figure 4: Plot of error versus iteration for AdaLISTA example. Here ISTA is the classic algorithm, AdaLISTA is the L2O operator in Algorithm 1 and Safe-L2O is the safeguarded version of AdaLISTA in Algorithm 2.



(a) Performance on *unseen* distribution

Figure 5: Plot of error versus iteration for AdaLISTA example. Here ISTA is the classic algorithm, AdaLISTA is the L2O operator in Algorithm 1 and Safe-L2O is the safeguarded version of AdaLISTA in Algorithm 2. 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 $k > 7$.

Conclusions

Numerous insights may be drawn from our examples. The first observation is, roughly speaking, each L2O scheme in our numerical examples reduces computational costs by at least one order of magnitude when applied to data from the same distribution as the training data (as compared to analytic optimization algorithms). This is consistent with results of previous works (*n.b.* on seen distributions, the safeguard is never triggered in our experiments, making the iterates for L2O and Safe-L2O identical). More importantly, plots in Figures 1b and 2b, and 5 show the safeguard steers updates to convergence when they would otherwise diverge or converge slower than the conventional algorithm. That is, Safe-L2O converges with data distinct from training while the nonsafeguarded L2O schemes diverge.

This work proposes a framework for ensuring convergence of L2O algorithms. Sequences generated by our Safe-L2O method provably converge to solutions of the optimization problems. Our Safe-L2O algorithm is also easy to implement as a wrapper around trained neural networks. 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

Aberdam, A.; Golts, A.; and Elad, M. 2021. Ada-lista: Learned solvers adaptive to varying models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

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

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

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

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

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

Chen, T.; Chen, X.; Chen, W.; Heaton, H.; Liu, J.; Wang, Z.; and Yin, W. 2022. Learning to Optimize: A Primer and A Benchmark. *Journal of Machine Learning Research*, 23(189): 1–59.

Chen, X.; Liu, J.; Wang, Z.; and Yin, W. 2018. 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*, 9061–9071. Curran Associates, Inc.

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

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

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

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

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

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

Liu, J.; Chen, X.; Wang, Z.; and Yin, W. 2019. ALISTA: Analytic Weights Are As Good As Learned Weights in LISTA. In *International Conference on Learning Representations*.

Mann, R. 1953. Mean Value Methods in Iteration. *Proceedings of the American Mathematical Society*, 4(3): 506–510.

Martin, D.; Fowlkes, C.; Tal, D.; and Malik, J. 2001. 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, 416–423.

Metzler, C.; Mousavi, A.; and Baraniuk, R. 2017. 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*, 1772–1783. Curran Associates, Inc.

Moreau, T.; and Bruna, J. 2017. Understanding Trainable Sparse Coding with Matrix Factorization. In *International Conference on Learning Representations*.

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

Themelis, A.; and Patrinos, P. 2019. SuperMann: a superlinearly convergent algorithm for finding fixed points of nonexpansive operators. *IEEE Transactions on Automatic Control*, 64(12): 4875–4890.

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

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

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

Xin, B.; Wang, Y.; Gao, W.; Wipf, D.; and Wang, B. 2016. 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*, 4340–4348. Curran Associates, Inc.

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

Yang, Y.; Sun, J.; Li, H.; and Xu, Z. 2016. 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*, 10–18. Curran Associates, Inc.

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