Towards Constituting Mathematical Structures for Learning to Optimize

Zhangyang "Atlas" Wang, UT Austin

(Most slides are generously shared by **Dr. Jialin Liu, Dr. Xiaohan Chen**, and **Dr. Wotao Yin**, Alibaba DAMO)

Outline

Outline

Learning to Optimize

Consider an optimization problem

$$\min_{\mathbf{x} \in \mathbb{R}^n} F(\mathbf{x})$$

Instead of manually designing an iterative algorithm

$$\mathbf{x}_{k+1} = \mathcal{T}_F(\mathbf{x}_k)$$

One may learn an update rule from data

$$\mathbf{x}_{k+1} = \mathcal{T}_F(\mathbf{x}_k; \theta)$$

where the parameter θ is obtained by minimizing a *loss function*

$$\min_{\theta \in \Theta} \mathbb{E}_{F \in \mathcal{F}} L(\mathbf{x}_K(\theta))$$

The set $\mathcal{F}$ consists of all instances of interest.

The process of minimizing the loss function is named *training*.

Such methodology is named *Learning to Optimize (L2O)*.

Examples

Example I: Learned ISTA (LISTA) [Gregor and LeCun, 2010]

- LASSO: $\mathcal{F} = \{(1/2)\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 + \lambda\|\mathbf{x}\|_1 : \mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m\}$
- Choose a baseline algorithm ISTA: $\mathbf{x}_{k+1} = \mathrm{prox}_{\theta_k}(\mathbf{x}_k - \alpha_k \mathbf{A}^\top(\mathbf{A}\mathbf{x}_k - \mathbf{b}))$
- Parameterization: $\mathbf{x}_{k+1} = \mathrm{prox}_{\theta_k}(\mathbf{W}_{1,k}\mathbf{x}_k + \mathbf{W}_{2,k}\mathbf{b})$

Example II: Learning a rule for step size [Xiong et al., 2022]

- Deep learning:
  $\mathcal{F} = \{f(\mathbf{x}) : f$ is the loss function of training neural networks$\}$
- Choose a baseline algorithm SGD: $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k$, where $\mathbf{g}_k$ is the stochastic gradient.
- Parameterization: $\alpha_k = \mathrm{NN}(\mathbf{x}_k, \mathbf{g}_k; \theta)$.

Examples

Example I: Learned ISTA (LISTA) [Gregor and LeCun, 2010]

- LASSO: $\mathcal{F} = \{(1/2)\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 + \lambda\|\mathbf{x}\|_1 : \mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m\}$
- Choose a baseline algorithm ISTA: $\mathbf{x}_{k+1} = \mathrm{prox}_{\theta_k}(\mathbf{x}_k - \alpha_k \mathbf{A}^\top(\mathbf{A}\mathbf{x}_k - \mathbf{b}))$
- Parameterization: $\mathbf{x}_{k+1} = \mathrm{prox}_{\theta_k}(\mathbf{W}_{1,k}\mathbf{x}_k + \mathbf{W}_{2,k}\mathbf{b})$

Example II: Learning a rule for step size [Xiong et al., 2022]

- Deep learning:
  $\mathcal{F} = \{f(\mathbf{x}) : f$ is the loss function of training neural networks$\}$
- Choose a baseline algorithm SGD: $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k$, where $\mathbf{g}_k$ is the stochastic gradient.
- Parameterization: $\alpha_k = \mathrm{NN}(\mathbf{x}_k, \mathbf{g}_k; \theta)$.

Sample instances from $\mathcal{F}$ and Learn an algorithm.

The learned algorithm works well on unseen instances in $\mathcal{F}$.

Discussions and Motivations

A tradeoff:

- A baseline algorithm works for a broad class of problems
- One may design advanced algorithms for specific algorithms

Discussions and Motivations

A tradeoff:

- A baseline algorithm works for a broad class of problems
- One may design advanced algorithms for specific algorithms

L2O provides a uniform tool to obtain customized algorithms without domain knowledge.

Discussions and Motivations

A tradeoff:

- A baseline algorithm works for a broad class of problems
- One may design advanced algorithms for specific algorithms

L2O provides a uniform tool to obtain customized algorithms without domain knowledge.

Questions:

- Can we find principles from learned algorithms?
- Can we use domain knowledge to regularize the models?

ML vs OPT

Machine learning (ML) is *induction*

- (problems, answers) are given for training
- ML learns to give answers in the future

Optimization (OPT) is *prescription*

- (problems, evaluations) are given, not answers
- OPT finds answers with best evaluations

Learning to optimize (L2O) combines ML and OPT to obtain “better” solutions “faster”, by learning from records of optimization.

Classic vs Learned

Classic OPT:

- Experts hand-built algorithms based on theory and experience
  For example, Simplex Method and Nesterov Accelerated Gradient Method
- Algorithms are written as iterations in a few lines
- Practitioners pick an algorithm to use

L2O:

- Experts propose L2O templates and training procedures
- Practitioners
  - pick an L2O template
  - prepare training data
  - apply a training procedure

  $\rightarrow$ obtain a trained algorithm for future problems
- Practitioners are more involved in the design process

Papers and Coauthors

This talk is based on the following articles:

- J. Liu, X. Chen, Z. Wang, W. Yin, and H. Cai. "Towards Constituting Mathematical Structures for Learning to Optimize." ICML 2023.
- X. Chen, J. Liu, Z. Wang, and W. Yin. "Hyperparameter Tuning is All You Need for LISTA." NeurIPS 2021.
- J. Liu, X. Chen, Z. Wang, and W. Yin. "ALISTA: Analytic weights are as good as learned weights in LISTA." ICLR 2019.
- X. Chen, J. Liu, Z. Wang, and W. Yin. "Theoretical Linear Convergence of Unfolded ISTA and its Practical Weights and Thresholds." NeurIPS 2018.

Outline

LASSO and ISTA

LASSO: assume $\mathbf{b} = \mathbf{A}\mathbf{x}_* +$ noise; recover $\mathbf{x}_*$ by solving

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_1$$

also known as ℓ_1-regularized least-squares and compressed sensing

LASSO and ISTA

LASSO: assume $\mathbf{b} = \mathbf{A}\mathbf{x}_* + \text{noise}$; recover $\mathbf{x}_*$ by solving

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_1$$

also known as ℓ_1-regularized least-squares and compressed sensing

Iterative soft-thresholding algorithm (ISTA):

$$\mathbf{x}_{k+1} = \eta_{\lambda\alpha}\left(\mathbf{x}_k - \alpha\mathbf{A}^\top(\mathbf{A}\mathbf{x}_k - \mathbf{b})\right)$$

- convergence requires a proper stepsize α or line search
- the gradient-descent step reduces $\frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$
- the soft-thresholding step $\eta_{\lambda\alpha}(\cdot)$ reduces $\lambda\|\mathbf{x}\|_1$

Learned ISTA [Gregor and LeCun, 2010]

Introduce scalar $\theta = \lambda\alpha$ and matrices $\mathbf{W}_1 = \alpha\mathbf{A}^\top$ and $\mathbf{W}_2 = \mathbf{I} - \alpha\mathbf{A}^\top\mathbf{A}$.

Rewrite ISTA as

$$\mathbf{x}_{k+1} = \eta_\theta(\mathbf{W}_1\mathbf{b} + \mathbf{W}_2\mathbf{x}_k).$$

Learned ISTA [Gregor and LeCun, 2010]

Introduce scalar $\theta = \lambda\alpha$ and matrices $\mathbf{W}_1 = \alpha\mathbf{A}^\top$ and $\mathbf{W}_2 = \mathbf{I} - \alpha\mathbf{A}^\top\mathbf{A}$.

Rewrite ISTA as

$$\mathbf{x}_{k+1} = \eta_\theta(\mathbf{W}_1\mathbf{b} + \mathbf{W}_2\mathbf{x}_k).$$

Introduce $\theta_k, \mathbf{W}_{1,k}, \mathbf{W}_{2,k}$, $k = 0, 1, \ldots, K-1$, as free parameters and define

$$\mathbf{x}_{k+1} = \eta_{\theta_k}(\mathbf{W}_{1,k}\mathbf{b} + \mathbf{W}_{2,k}\mathbf{x}_k), \quad k = 0, 1, \cdots, K-1.$$

Once $\{\theta_k, \mathbf{W}_{1,k}, \mathbf{W}_{2,k}\}_{k=0}^{K-1}$ are determined, we obtain a new algorithm.

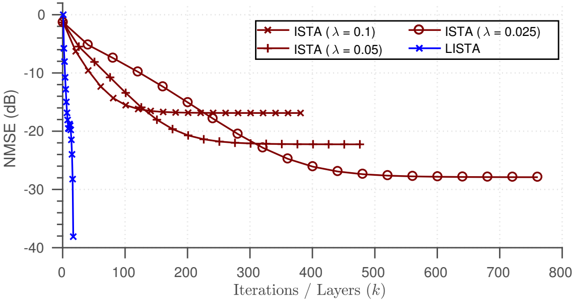
Find parameters such that the algorithm converges very fast for a set of LASSO instances with the same $\mathbf{A}$.

Fix random matrix $\mathbf{A}$, generate a set of sparse $\mathbf{x}_{*,i}$, with varying supports, and $\mathbf{b}_i = \mathbf{A}\mathbf{x}_{*,i} + \text{noise}_i$. Form the training set $\mathcal{F} = \{(\mathbf{x}_{*,i}, \mathbf{b}_i)\}$.

Fix a small $K > 0$, and train the parameters by applying SGD to

$$\min_{\{\theta_k, \mathbf{W}_{1,k}, \mathbf{W}_{2,k}\}_{k=0}^{K-1}} \mathbb{E}_{(\mathbf{x}_*, \mathbf{b}) \in \mathcal{F}} \left\| \mathbf{x}_K(\mathbf{b}) - \mathbf{x}_* \right\|_2^2 .$$

After the NN is trained with $K = 16$:



The trained NN is called Learned ISTA (LISTA).

Weight coupling

Given the superb performance,

can we find some principles from the learned algorithm?

Weight coupling

Given the superb performance,
can we find some principles from the learned algorithm?

Suppose the learned algorithm is an "ideal" algorithm: exactly recover $\mathbf{x}_*$ given infinite many steps.

Weight coupling

Given the superb performance,
can we find some principles from the learned algorithm?

Suppose the learned algorithm is an "ideal" algorithm: exactly recover $\mathbf{x}_*$ given infinite many steps.

Theorem

Assume no noise. If LISTA has $\mathbf{x}_k \to \mathbf{x}_*$ as $k \to \infty$ uniformly for all sparse $\mathbf{x}_*$, then the parameters $\{\theta_k, \mathbf{W}_{1,k}, \mathbf{W}_{2,k}\}_{k=0}^{\infty}$ must satisfy the relation

$$\mathbf{W}_{2,k} + \mathbf{W}_{1,k}\mathbf{A} \to \mathbf{I}, \quad \text{as } k \to \infty.$$

Weight coupling

Given the superb performance,
can we find some principles from the learned algorithm?

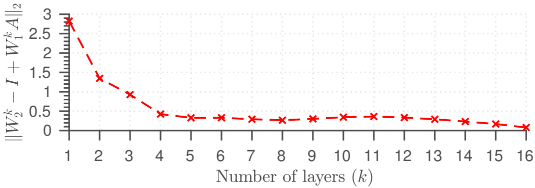
Suppose the learned algorithm is an "ideal" algorithm: exactly recover $\mathbf{x}_*$ given infinite many steps.

Theorem

Assume no noise. If LISTA has $\mathbf{x}_k \to \mathbf{x}_*$ as $k \to \infty$ uniformly for all sparse $\mathbf{x}_*$, then the parameters $\{\theta_k, \mathbf{W}_{1,k}, \mathbf{W}_{2,k}\}_{k=0}^{\infty}$ must satisfy the relation

$$\mathbf{W}_{2,k} + \mathbf{W}_{1,k}\mathbf{A} \to \mathbf{I}, \quad \text{as } k \to \infty.$$

Indeed, training confirms the claims:



Therefore, we enforce

$$\mathbf{W}_{2,k} = \mathbf{I} - \mathbf{W}_{1,k}\mathbf{A},$$

for all k, yielding the iteration:

$$\mathbf{x}_{k+1} = \eta_{\theta_k}(\mathbf{x}_k + \mathbf{W}_{1,k}(\mathbf{b} - \mathbf{A}\mathbf{x}_k)).$$

We call it *weight coupling (CP).*

Therefore, we enforce

$$\mathbf{W}_{2,k} = \mathbf{I} - \mathbf{W}_{1,k}\mathbf{A},$$

for all k, yielding the iteration:

$$\mathbf{x}_{k+1} = \eta_{\theta_k}(\mathbf{x}_k + \mathbf{W}_{1,k}(\mathbf{b} - \mathbf{A}\mathbf{x}_k)).$$

We call it *weight coupling (CP).*

Parameters

$$\mathcal{O}(n^2K + mnK) \xrightarrow{\text{reduce}} \mathcal{O}(mnK),$$

significant reduction if $m < n$ (which is often the case).

After this reduction, training also appears to be more stable.

Empirical Settings

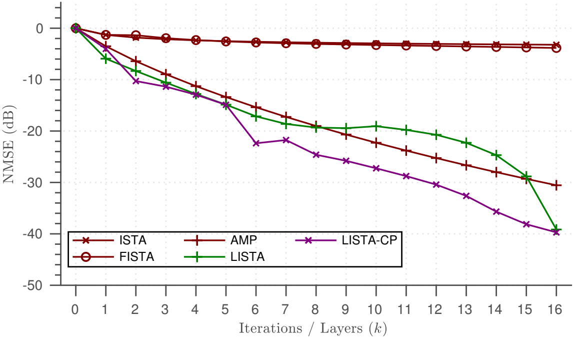
Normalized MSE (NMSE) in dB:

$$\mathrm{NMSE}(\hat{\mathbf{x}}, \mathbf{x}_*) = 20 \log_{10} \left(\|\hat{\mathbf{x}} - \mathbf{x}_*\|_2 / \|\mathbf{x}_*\|_2 \right)$$

Tests:

- $m = 250$, $n = 500$, sparsity $s \approx 50$.
- $\mathbf{A}_{ij} \sim \mathcal{N}(0, 1/\sqrt{m})$, iid. $\mathbf{A}$ is column-normalized.
- Magnitudes were sampled from standard Gaussian.

Weight coupling (CP)



CP stabilizes intermediate results.

Same final recovery quality.

Outline

A general L2O model

Consider $\min_{\mathbf{x}\in\mathbb{R}^n} F(\mathbf{x})$.

A baseline manually designed algorithm: gradient descent with momentum:

$$\begin{aligned}\mathbf{v}_{k+1} =& \beta_k \mathbf{v}_k + (1-\beta_k)\nabla F(\mathbf{x}_k),\\ \mathbf{x}_{k+1} =& \mathbf{x}_k - \alpha_k \mathbf{v}_{k+1}, \qquad k = 0,1,2,\ldots\end{aligned}$$

Andrychowicz et al. [2016] proposed to learn a parameterized algorithm:

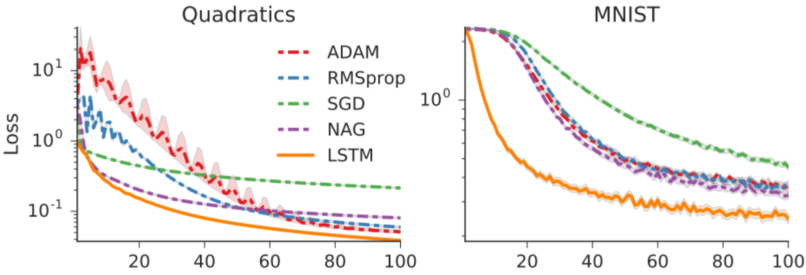
$$\begin{aligned}\mathbf{d}_k, \mathbf{h}_k =& \mathrm{LSTM}\Big(\mathbf{x}_k, \nabla F(\mathbf{x}_k), \mathbf{h}_{k-1}; \phi\Big)\\ \mathbf{x}_{k+1} =& \mathbf{x}_k - \mathbf{d}_k\end{aligned}$$

by minimizing a loss function

$$\min_{\phi} \mathbb{E}_{F\in\mathcal{F}} \sum_{k=1}^{K} F(\mathbf{x}_k)$$

Term “LSTM” means a long short-term memory cell.

Numerical results



Some discussions

Observation: The learned update rule may diverge on unseen instances.
This is still an active topic in the literature. [Wichrowska et al., 2017, Wu et al., 2018, Metz et al., 2019, Chen et al., 2020, Harrison et al., 2022, Metz et al., 2022]

Some discussions

Observation: The learned update rule may diverge on unseen instances. This is still an active topic in the literature. [Wichrowska et al., 2017, Wu et al., 2018, Metz et al., 2019, Chen et al., 2020, Harrison et al., 2022, Metz et al., 2022]

Question: Can we find those conditions that $\mathbf{d}_k$ should satisfy if we assume $\mathbf{x}_k \to \mathbf{x}_*$?

Some discussions

Observation: The learned update rule may diverge on unseen instances.
This is still an active topic in the literature. [Wichrowska et al., 2017, Wu et al., 2018, Metz et al., 2019, Chen et al., 2020, Harrison et al., 2022, Metz et al., 2022]

Question: Can we find those conditions that $\mathbf{d}_k$ should satisfy if we assume $\mathbf{x}_k \to \mathbf{x}_*$?

Preparations:

- Assumptions on the objective function F:

(Smooth case) $F(\mathbf{x}) = f(\mathbf{x})$, where f is convex and differentiable with Lipschitz continuous gradient
(Nonsmooth case) $F(\mathbf{x}) = r(\mathbf{x})$, where r is proper, closed and convex.
(Composite case) $F(\mathbf{x}) = f(\mathbf{x}) + r(\mathbf{x})$

Some discussions

Observation: The learned update rule may diverge on unseen instances.
This is still an active topic in the literature. [Wichrowska et al., 2017, Wu et al., 2018, Metz et al., 2019, Chen et al., 2020, Harrison et al., 2022, Metz et al., 2022]

Question: Can we find those conditions that $\mathbf{d}_k$ should satisfy if we assume $\mathbf{x}_k \to \mathbf{x}_*$?

Preparations:

- Assumptions on the objective function F:

(Smooth case) $F(\mathbf{x}) = f(\mathbf{x})$, where f is convex and differentiable with Lipschitz continuous gradient
(Nonsmooth case) $F(\mathbf{x}) = r(\mathbf{x})$, where r is proper, closed and convex.
(Composite case) $F(\mathbf{x}) = f(\mathbf{x}) + r(\mathbf{x})$

- Assumptions on the update direction $\{\mathbf{d}_k\}$

Basic settings for smooth case

The update direction $\mathbf{d}_k$ is generated by $\mathrm{LSTM}\big(\mathbf{x}_k, \nabla f(\mathbf{x}_k), \mathbf{h}_{k-1}; \phi\big)$

Basic settings for smooth case

The update direction $\mathbf{d}_k$ is generated by $\mathrm{LSTM}\Big(\mathbf{x}_k, \nabla f(\mathbf{x}_k), \mathbf{h}_{k-1}; \phi\Big)$

Write $\mathbf{d}_k = \mathbf{m}\Big(\mathbf{x}_k, \nabla f(\mathbf{x}_k), \mathbf{h}_{k-1}; \phi\Big)$ where $\mathbf{m}$ is a parameterized operator

Basic settings for smooth case

The update direction $\mathbf{d}_k$ is generated by $\mathrm{LSTM}\Big(\mathbf{x}_k, \nabla f(\mathbf{x}_k), \mathbf{h}_{k-1}; \phi\Big)$

Write $\mathbf{d}_k = \mathbf{m}\Big(\mathbf{x}_k, \nabla f(\mathbf{x}_k), \mathbf{h}_{k-1}; \phi\Big)$ where $\mathbf{m}$ is a parameterized operator

With $\mathbf{m}_k(\cdot, \cdot) := \mathbf{m}(\cdot, \cdot, \mathbf{h}_{k-1})$, we write $\mathbf{d}_k = \mathbf{m}_k\Big(\mathbf{x}_k, \nabla f(\mathbf{x}_k); \phi\Big)$

Basic settings for smooth case

The update direction $\mathbf{d}_k$ is generated by $\mathrm{LSTM}\Big(\mathbf{x}_k, \nabla f(\mathbf{x}_k), \mathbf{h}_{k-1}; \phi\Big)$

Write $\mathbf{d}_k = \mathbf{m}\Big(\mathbf{x}_k, \nabla f(\mathbf{x}_k), \mathbf{h}_{k-1}; \phi\Big)$ where $\mathbf{m}$ is a parameterized operator

With $\mathbf{m}_k(\cdot,\cdot) := \mathbf{m}(\cdot,\cdot,\mathbf{h}_{k-1})$, we write $\mathbf{d}_k = \mathbf{m}_k\Big(\mathbf{x}_k, \nabla f(\mathbf{x}_k); \phi\Big)$

Let's consider a more general update rule

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{d}_k(\mathbf{x}_k, \nabla f(\mathbf{x}_k))$$

where $\mathbf{d}_k$ is an operator picked from

$$\mathcal{D}_C(\mathbb{R}^{2n}) = \Big\{\mathbf{d} : \mathbb{R}^{2n} \to \mathbb{R}^n \;\Big|\; \mathbf{d} \text{ is differentiable, } \|\mathrm{J}\mathbf{d}(\mathbf{z})\|_{\mathrm{F}} \le C,\ \forall \mathbf{z} \in \mathcal{Z}\Big\}.$$

- Training needs derivatives of $\mathbf{d}_k$.
- Many existing parameterization approaches yield $\mathbf{d}_k \in \mathcal{D}_C(\mathbb{R}^{2n})$.

Core Assumptions

What conditions should the update rule follow?

Core Assumptions

What conditions should the update rule follow?

- (Global Convergence) For any sequences $\{\mathbf{x}_k\}_{k=0}^{\infty}$ generated by the given rule, there exists $\mathbf{x}_* \in \arg\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$ such that $\lim_{k\to\infty} \mathbf{x}_k = \mathbf{x}_*$.

Core Assumptions

What conditions should the update rule follow?

- (Global Convergence) For any sequences $\{\mathbf{x}_k\}_{k=0}^{\infty}$ generated by the given rule, there exists $\mathbf{x}_* \in \arg\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$ such that $\lim_{k\to\infty} \mathbf{x}_k = \mathbf{x}_*$.

Fixed point assumption: $\mathbf{x}_{k+1} = \mathbf{x}_*$ as long as $\mathbf{x}_k = \mathbf{x}_*$:

$$\mathbf{x}_* = \mathbf{x}_* - \mathbf{d}_k(\mathbf{x}_*, \nabla f(\mathbf{x}_*))$$

Core Assumptions

What conditions should the update rule follow?

- (Global Convergence) For any sequences $\{\mathbf{x}_k\}_{k=0}^{\infty}$ generated by the given rule, there exists $\mathbf{x}_* \in \arg\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$ such that $\lim_{k\to\infty} \mathbf{x}_k = \mathbf{x}_*$.

Fixed point assumption: $\mathbf{x}_{k+1} = \mathbf{x}_*$ as long as $\mathbf{x}_k = \mathbf{x}_*$:

$$\mathbf{x}_* = \mathbf{x}_* - \mathbf{d}_k(\mathbf{x}_*, \nabla f(\mathbf{x}_*))$$

Convex analysis theory tells us $\nabla f(\mathbf{x}_*) = \mathbf{0}$, and we obtain $\mathbf{d}_k(\mathbf{x}_*, \mathbf{0}) = \mathbf{0}$.

Core Assumptions

What conditions should the update rule follow?

- (Global Convergence) For any sequences $\{\mathbf{x}_k\}_{k=0}^{\infty}$ generated by the given rule, there exists $\mathbf{x}_* \in \arg\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$ such that $\lim_{k\to\infty} \mathbf{x}_k = \mathbf{x}_*$.

Fixed point assumption: $\mathbf{x}_{k+1} = \mathbf{x}_*$ as long as $\mathbf{x}_k = \mathbf{x}_*$:

$$\mathbf{x}_* = \mathbf{x}_* - \mathbf{d}_k(\mathbf{x}_*, \nabla f(\mathbf{x}_*))$$

Convex analysis theory tells us $\nabla f(\mathbf{x}_*) = \mathbf{0}$, and we obtain $\mathbf{d}_k(\mathbf{x}_*, \mathbf{0}) = \mathbf{0}$.

- (Asympototic Fixed Point Condition) Formally, we relax it and assume

$$\lim_{k\to\infty} \mathbf{d}_k(\mathbf{x}_*, \mathbf{0}) = \mathbf{0}$$

for any $\mathbf{x}_* \in \arg\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$.

Core Assumptions

What conditions should the update rule follow?

- (Global Convergence) For any sequences $\{\mathbf{x}_k\}_{k=0}^{\infty}$ generated by the given rule, there exists $\mathbf{x}_* \in \arg\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$ such that $\lim_{k\to\infty} \mathbf{x}_k = \mathbf{x}_*$.

Fixed point assumption: $\mathbf{x}_{k+1} = \mathbf{x}_*$ as long as $\mathbf{x}_k = \mathbf{x}_*$:

$$\mathbf{x}_* = \mathbf{x}_* - \mathbf{d}_k(\mathbf{x}_*, \nabla f(\mathbf{x}_*))$$

Convex analysis theory tells us $\nabla f(\mathbf{x}_*) = \mathbf{0}$, and we obtain $\mathbf{d}_k(\mathbf{x}_*, \mathbf{0}) = \mathbf{0}$.

- (Asympototic Fixed Point Condition) Formally, we relax it and assume

$$\lim_{k\to\infty} \mathbf{d}_k(\mathbf{x}_*, \mathbf{0}) = \mathbf{0}$$

for any $\mathbf{x}_* \in \arg\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$.

The two assumptions are coined as *(GC)* and *(FP)*, respectively.

A Preliminary Result

Theorem

For any f and any operator sequence $\{\mathbf{d}_k\}_{k=0}^{\infty}$ that satisfies (GC) and (FP), there exist $\mathbf{P}_k \in \mathbb{R}^{n\times n}$ and $\mathbf{b}_k \in \mathbb{R}^n$ satisfying

$$\mathbf{d}_k(\mathbf{x}_k, \nabla f(\mathbf{x}_k)) = \mathbf{P}_k \nabla f(\mathbf{x}_k) + \mathbf{b}_k,$$

with $\mathbf{P}_k$ is bounded and $\mathbf{b}_k \to \mathbf{0}$ as $k \to \infty$.

A Preliminary Result

Theorem

For any f and any operator sequence $\{\mathbf{d}_k\}_{k=0}^{\infty}$ that satisfies (GC) and (FP), there exist $\mathbf{P}_k \in \mathbb{R}^{n\times n}$ and $\mathbf{b}_k \in \mathbb{R}^n$ satisfying

$$\mathbf{d}_k(\mathbf{x}_k, \nabla f(\mathbf{x}_k)) = \mathbf{P}_k \nabla f(\mathbf{x}_k) + \mathbf{b}_k,$$

with $\mathbf{P}_k$ is bounded and $\mathbf{b}_k \to \mathbf{0}$ as $k \to \infty$.

- A "good" update rule is not totally free.

A Preliminary Result

Theorem

For any f and any operator sequence $\{\mathbf{d}_k\}_{k=0}^{\infty}$ that satisfies (GC) and (FP), there exist $\mathbf{P}_k \in \mathbb{R}^{n\times n}$ and $\mathbf{b}_k \in \mathbb{R}^n$ satisfying

$$\mathbf{d}_k(\mathbf{x}_k, \nabla f(\mathbf{x}_k)) = \mathbf{P}_k \nabla f(\mathbf{x}_k) + \mathbf{b}_k,$$

with $\mathbf{P}_k$ is bounded and $\mathbf{b}_k \to \mathbf{0}$ as $k \to \infty$.

- A "good" update rule is not totally free.
- It covers many optimization algorithms, such as accelerated GD, quasi-Newton methods, etc.

A Preliminary Result

Theorem

For any f and any operator sequence $\{\mathbf{d}_k\}_{k=0}^{\infty}$ that satisfies (GC) and (FP), there exist $\mathbf{P}_k \in \mathbb{R}^{n\times n}$ and $\mathbf{b}_k \in \mathbb{R}^n$ satisfying

$$\mathbf{d}_k(\mathbf{x}_k, \nabla f(\mathbf{x}_k)) = \mathbf{P}_k \nabla f(\mathbf{x}_k) + \mathbf{b}_k,$$

with $\mathbf{P}_k$ is bounded and $\mathbf{b}_k \to \mathbf{0}$ as $k \to \infty$.

- A "good" update rule is not totally free.
- It covers many optimization algorithms, such as accelerated GD, quasi-Newton methods, etc.
- Instead of learning $\mathbf{d}_k$, one may learn a *preconditioner* $\mathbf{P}_k$ and a *bias* $\mathbf{b}_k$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{P}_k\big(\mathbf{x}_k; \phi\big)\nabla f(\mathbf{x}_k) - \mathbf{b}_k\big(\mathbf{x}_k; \psi\big),$$

Nonsmooth case

On nonsmooth problems $\min_{\mathbf{x}} r(\mathbf{x})$, a direct extension to gradient descent is sub-gradient descent: $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k,\ \mathbf{g}_k \in \partial r(\mathbf{x}_k)$.

Nonsmooth case

On nonsmooth problems $\min_{\mathbf{x}} r(\mathbf{x})$, a direct extension to gradient descent is sub-gradient descent: $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k,\ \mathbf{g}_k \in \partial r(\mathbf{x}_k)$.

Such explicit rule suffers from convergence issues.

Nonsmooth case

On nonsmooth problems $\min_{\mathbf{x}} r(\mathbf{x})$, a direct extension to gradient descent is sub-gradient descent: $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k,\ \mathbf{g}_k \in \partial r(\mathbf{x}_k)$.

Such explicit rule suffers from convergence issues.

An implicit rule like proximal point algorithm (PPA) converges much better:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_{k+1}, \quad \mathbf{g}_{k+1} \in \partial r(\mathbf{x}_{k+1}).$$

Nonsmooth case

On nonsmooth problems $\min_{\mathbf{x}} r(\mathbf{x})$, a direct extension to gradient descent is sub-gradient descent: $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k,\ \mathbf{g}_k \in \partial r(\mathbf{x}_k)$.

Such explicit rule suffers from convergence issues.

An implicit rule like proximal point algorithm (PPA) converges much better:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_{k+1}, \quad \mathbf{g}_{k+1} \in \partial r(\mathbf{x}_{k+1}).$$

Back to L2O, we choose an implicit rule:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{d}_k(\mathbf{x}_{k+1}, \mathbf{g}_{k+1}), \quad \mathbf{g}_{k+1} \in \partial r(\mathbf{x}_{k+1}).$$

Implicit rule:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{d}_k(\mathbf{x}_{k+1}, \mathbf{g}_{k+1}), \quad \mathbf{g}_{k+1} \in \partial r(\mathbf{x}_{k+1}). \tag{1}$$

Theorem

For each r and any $\{\mathbf{d}_k\}_{k=0}^{\infty}$ that satisfies (GC) and (FP), there exist $\mathbf{P}_k \in \mathbb{R}^{n\times n}$ and $\mathbf{b}_k \in \mathbb{R}^n$ such that (1) *yields*

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{P}_k \mathbf{g}_{k+1} - \mathbf{b}_k, \quad \mathbf{g}_{k+1} \in \partial r(\mathbf{x}_{k+1}),$$

with $\mathbf{P}_k$ is bounded and $\mathbf{b}_k \to \mathbf{0}$ as $k \to \infty$. If we further assume $\mathbf{P}_k \succ \mathbf{0}$, $\mathbf{x}_{k+1}$ can be uniquely determined through $\mathbf{x}_{k+1} = \mathbf{prox}_{r,\mathbf{P}_k}(\mathbf{x}_k - \mathbf{b}_k)$.

The proximal operator $\mathbf{prox}_{r,\mathbf{P}_k}$ is defined with $\mathbf{prox}_{r,\mathbf{P}}(\bar{\mathbf{x}}) := \arg\min_{\mathbf{x}} r(\mathbf{x}) + \frac{1}{2}\|\mathbf{x}-\bar{\mathbf{x}}\|^2_{\mathbf{P}^{-1}}$.

- Global Convergence and Asymptotic Fixed Point Condition imply (1) yields a structure.
- A generalized proximal point algorithm. Fix $\mathbf{P}_k = \alpha\mathbf{I}, \mathbf{b}_k = \mathbf{0}$, it reduces to PPA.

Composite Case

Consider the composite case $\min_{\mathbf{x}} f(\mathbf{x}) + r(\mathbf{x})$. We analyze a mixed rule

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{d}_k(\mathbf{x}_k, \nabla f(\mathbf{x}_k), \mathbf{x}_{k+1}, \mathbf{g}_{k+1}), \quad \mathbf{g}_{k+1} \in \partial r(\mathbf{x}_{k+1}). \tag{2}$$

Theorem

For any $f, r, \{\mathbf{d}_k\}_{k=0}^{\infty}$ that satisfies (GC) and (FP), there exist $\mathbf{P}_k \in \mathbb{R}^{n\times n}$ and $\mathbf{b}_k \in \mathbb{R}^n$ such that (2) *yields*

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{P}_k(\nabla f(\mathbf{x}_k) - \mathbf{g}_{k+1}) - \mathbf{b}_k, \;\; \mathbf{g}_{k+1} \in \partial r(\mathbf{x}_{k+1}),$$

with $\mathbf{P}_k$ is bounded and $\mathbf{b}_k \to \mathbf{0}$ as $k \to \infty$. If we further assume $\mathbf{P}_k \succ \mathbf{0}$, $\mathbf{x}_{k+1}$ can be uniquely determined given $\mathbf{x}_k$ through

$$\mathbf{x}_{k+1} = \mathbf{prox}_{r,\mathbf{P}_k}(\mathbf{x}_k - \mathbf{P}_k \nabla f(\mathbf{x}_k) - \mathbf{b}_k). \tag{3}$$

With $\mathbf{P}_k = \alpha \mathbf{I}, \mathbf{b}_k = \mathbf{0}$, (3) reduces to Proximal Gradient Descent (PGD).

Longer Horizen

Introduce an extra variable $\mathbf{y}_k$ that encodes historical information

$$\mathbf{y}_k = \mathbf{m}(\mathbf{x}_k, \mathbf{x}_{k-1}, \cdots, \mathbf{x}_{k-T}).$$

Insert $\mathbf{y}_k$ to the previous update rule

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{d}_k(\mathbf{x}_k, \nabla f(\mathbf{x}_k), \mathbf{x}_{k+1}, \mathbf{g}_{k+1}, \mathbf{y}_k, \nabla f(\mathbf{y}_k)), \quad \mathbf{g}_{k+1} \in \partial r(\mathbf{x}_{k+1})$$

Theorem

Suppose $T = 1$. For any $f, r, \mathbf{m}, \{\mathbf{d}_k\}_{k=0}^{\infty}$ that satisfies (GC) and (FP), there exist $\mathbf{P}_{1,k}, \mathbf{P}_{2,k}, \mathbf{A}_k \in \mathbb{R}^{n\times n}$ and $\mathbf{b}_{1,k}, \mathbf{b}_{2,k} \in \mathbb{R}^n$ satisfying

$$\begin{aligned}
\mathbf{x}_{k+1} &= \mathbf{x}_k - (\mathbf{P}_{1,k} - \mathbf{P}_{2,k})\nabla f(\mathbf{x}_k) - \mathbf{P}_{2,k}\nabla f(\mathbf{y}_k) - \mathbf{b}_{1,k} \\
&\qquad - \mathbf{P}_{1,k}\mathbf{g}_{k+1} - \mathbf{B}_k(\mathbf{y}_k - \mathbf{x}_k), \ \mathbf{g}_{k+1} \in \partial r(\mathbf{x}_{k+1}), \\
\mathbf{y}_{k+1} &= (\mathbf{I} - \mathbf{A}_k)\mathbf{x}_{k+1} + \mathbf{A}_k\mathbf{x}_k + \mathbf{b}_{2,k}
\end{aligned}$$

for all $k = 0, 1, 2, \cdots$, with $\{\mathbf{P}_{1,k}, \mathbf{P}_{2,k}, \mathbf{A}_k\}$ bounded and $\mathbf{b}_{1,k} \to \mathbf{0}, \mathbf{b}_{2,k} \to \mathbf{0}$ as $k \to \infty$.

L2O Model and Parameterization

If we further assume $\mathbf{P}_{1,k}$ is uniformly symmetric positive definite, then we can substitute $\mathbf{P}_{2,k}\mathbf{P}_{1,k}^{-1}$ with $\mathbf{B}_k$ and obtain

$$\begin{aligned}
\hat{\mathbf{x}}_k &= \mathbf{x}_k - \mathbf{P}_{1,k}\nabla f(\mathbf{x}_k), \\
\hat{\mathbf{y}}_k &= \mathbf{y}_k - \mathbf{P}_{1,k}\nabla f(\mathbf{y}_k), \\
\mathbf{x}_{k+1} &= \mathbf{prox}_{r,\mathbf{P}_{1,k}}\Big((\mathbf{I}-\mathbf{B}_k)\hat{\mathbf{x}}_k + \mathbf{B}_k\hat{\mathbf{y}}_k - \mathbf{b}_{1,k}\Big), \\
\mathbf{y}_{k+1} &= \mathbf{x}_{k+1} + \mathbf{A}_k(\mathbf{x}_{k+1}-\mathbf{x}_k) + \mathbf{b}_{2,k}.
\end{aligned}$$

L2O Model and Parameterization

If we further assume $\mathbf{P}_{1,k}$ is uniformly symmetric positive definite, then we can substitute $\mathbf{P}_{2,k}\mathbf{P}_{1,k}^{-1}$ with $\mathbf{B}_k$ and obtain

$$\begin{aligned}
\hat{\mathbf{x}}_k &= \mathbf{x}_k - \mathbf{P}_{1,k}\nabla f(\mathbf{x}_k), \\
\hat{\mathbf{y}}_k &= \mathbf{y}_k - \mathbf{P}_{1,k}\nabla f(\mathbf{y}_k), \\
\mathbf{x}_{k+1} &= \mathbf{prox}_{r,\mathbf{P}_{1,k}}\Big((\mathbf{I}-\mathbf{B}_k)\hat{\mathbf{x}}_k + \mathbf{B}_k\hat{\mathbf{y}}_k - \mathbf{b}_{1,k}\Big), \\
\mathbf{y}_{k+1} &= \mathbf{x}_{k+1} + \mathbf{A}_k(\mathbf{x}_{k+1} - \mathbf{x}_k) + \mathbf{b}_{2,k}.
\end{aligned}$$

We suggest using diagonal matrices for $\mathbf{P}_{1,k}, \mathbf{B}_k, \mathbf{A}_k$ in practice:

$$\mathbf{P}_{1,k} = \mathrm{diag}(\mathbf{p}_k), \quad \mathbf{B}_k = \mathrm{diag}(\mathbf{b}_k), \quad \mathbf{A}_k = \mathrm{diag}(\mathbf{a}_k),$$

where $\mathbf{p}_k, \mathbf{b}_k, \mathbf{a}_k \in \mathbb{R}^n$ are vectors.

L2O Model and Parameterization

If we further assume $\mathbf{P}_{1,k}$ is uniformly symmetric positive definite, then we can substitute $\mathbf{P}_{2,k}\mathbf{P}_{1,k}^{-1}$ with $\mathbf{B}_k$ and obtain

$$
\begin{aligned}
\hat{\mathbf{x}}_k &= \mathbf{x}_k - \mathbf{P}_{1,k}\nabla f(\mathbf{x}_k),\\
\hat{\mathbf{y}}_k &= \mathbf{y}_k - \mathbf{P}_{1,k}\nabla f(\mathbf{y}_k),\\
\mathbf{x}_{k+1} &= \mathbf{prox}_{r,\mathbf{P}_{1,k}}\Big((\mathbf{I}-\mathbf{B}_k)\hat{\mathbf{x}}_k + \mathbf{B}_k\hat{\mathbf{y}}_k - \mathbf{b}_{1,k}\Big),\\
\mathbf{y}_{k+1} &= \mathbf{x}_{k+1} + \mathbf{A}_k(\mathbf{x}_{k+1}-\mathbf{x}_k) + \mathbf{b}_{2,k}.
\end{aligned}
$$

We suggest using diagonal matrices for $\mathbf{P}_{1,k}, \mathbf{B}_k, \mathbf{A}_k$ in practice:

$$
\mathbf{P}_{1,k} = \mathrm{diag}(\mathbf{p}_k), \quad \mathbf{B}_k = \mathrm{diag}(\mathbf{b}_k), \quad \mathbf{A}_k = \mathrm{diag}(\mathbf{a}_k),
$$

where $\mathbf{p}_k, \mathbf{b}_k, \mathbf{a}_k \in \mathbb{R}^n$ are vectors.

We model $\mathbf{p}_k$, $\mathbf{a}_k$, $\mathbf{b}_k$, $\mathbf{b}_{1,k}$, $\mathbf{b}_{2,k}$ as the output of LSTM:

$$
\begin{aligned}
\mathbf{o}_k, \mathbf{h}_k &= \mathrm{LSTM}\Big(\mathbf{x}_k, \nabla f(\mathbf{x}_k), \mathbf{h}_{k-1}; \phi_{\mathsf{LSTM}}\Big),\\
\mathbf{p}_k, \mathbf{a}_k, \mathbf{b}_k, \mathbf{b}_{1,k}, \mathbf{b}_{2,k} &= \mathrm{MLP}(\mathbf{o}_k; \phi_{\mathsf{MLP}}).
\end{aligned}
$$

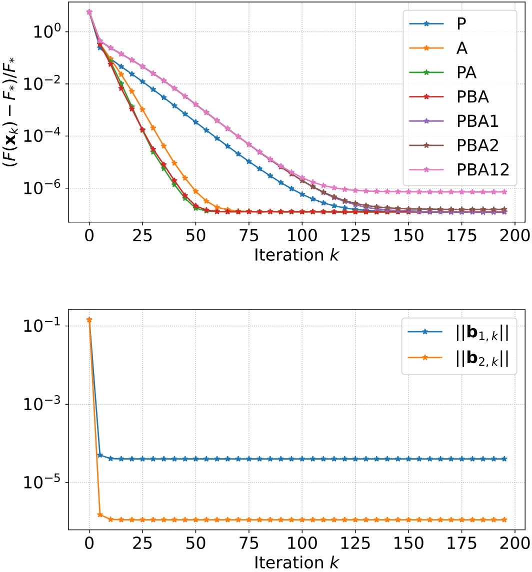
Ablation Study

We compare

- **PBA12**: $\mathbf{p}_k$,$\mathbf{a}_k$,$\mathbf{b}_k$,$\mathbf{b}_{1,k}$,$\mathbf{b}_{2,k}$ are all learnable.
- **PBA1**: $\mathbf{p}_k$,$\mathbf{a}_k$,$\mathbf{b}_k$,$\mathbf{b}_{1,k}$ are learnable; $\mathbf{b}_{2,k} = \mathbf{0}$.
- **PBA2**: $\mathbf{p}_k$,$\mathbf{a}_k$,$\mathbf{b}_k$,$\mathbf{b}_{2,k}$ are learnable; $\mathbf{b}_{1,k} = \mathbf{0}$.
- **PBA**: $\mathbf{p}_k$,$\mathbf{a}_k$,$\mathbf{b}_k$ are learnable; $\mathbf{b}_{2,k} = \mathbf{b}_{1,k} = \mathbf{0}$.
- **PA**: $\mathbf{p}_k$,$\mathbf{a}_k$ are learnable; $\mathbf{b}_{2,k} = \mathbf{b}_{1,k} = \mathbf{0}$; $\mathbf{b}_k = \mathbf{1}$.
- **P**: only $\mathbf{p}_k$ is learnable; $\mathbf{a}_k = \mathbf{b}_{2,k} = \mathbf{b}_{1,k} = \mathbf{0}$; $\mathbf{b}_k = \mathbf{1}$.
- **A**: only $\mathbf{a}_k$ is learnable; $\mathbf{b}_{2,k} = \mathbf{b}_{1,k} = \mathbf{0}$; $\mathbf{b}_k = \mathbf{1}$; $\mathbf{p}_k = (1/L)\mathbf{1}$.

on more challenging LASSO settings: $\mathbf{A}$ is not fixed; each LASSO instance takes an independently generated $\mathbf{A}$.

Ablation study: Results



Final model

We adopt (**PA**) and fix $\mathbf{b}_{1,k} = \mathbf{b}_{2,k} = \mathbf{0}$ and $\mathbf{b}_k = \mathbf{1}$.

$$
\begin{aligned}
\mathbf{o}_k, \mathbf{h}_k &= \mathrm{LSTM}\Big(\mathbf{x}_k, \nabla f(\mathbf{x}_k), \mathbf{h}_{k-1}; \phi_{\mathsf{LSTM}}\Big), \\
\mathbf{p}_k, \mathbf{a}_k &= \mathrm{MLP}(\mathbf{o}_k; \phi_{\mathsf{MLP}}), \\
\mathbf{x}_{k+1} &= \mathbf{prox}_{r,\mathbf{p}_k}\Big(\mathbf{y}_k - \mathbf{p}_k \odot \nabla f(\mathbf{y}_k)\Big), \\
\mathbf{y}_{k+1} &= \mathbf{x}_{k+1} + \mathbf{a}_k \odot (\mathbf{x}_{k+1} - \mathbf{x}_k).
\end{aligned}
$$

Instead of learning the update rule, we suggest learning a preconditioner $\mathbf{p}_k$ and an accelerator $\mathbf{a}_k$.

Comparison: In-Distribution Test

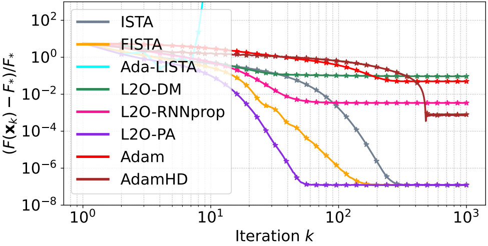


Figure: LASSO: Train and test on synthetic data.

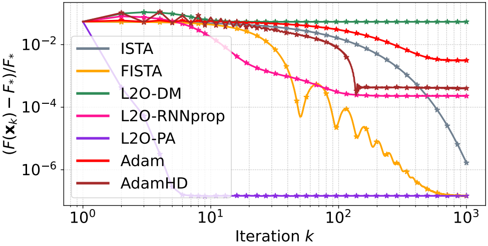


Figure: Logistic: Train and test on synthetic data.

Comparison: Out-of-Distribution Test

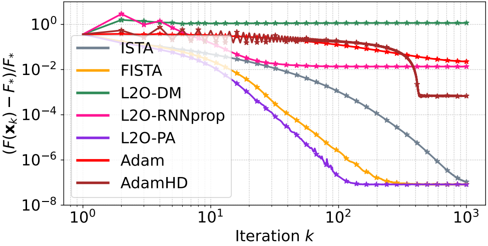


Figure: LASSO: Train on synthetic data and test on real data (BSDS500).

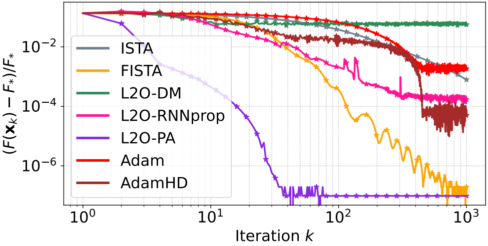


Figure: Logistic: Train on synthetic data and test on real data (Ionosphere).

Outline

Further analysis

Recall LISTA-CP model:

$$\mathbf{x}_{k+1} = \eta_{\theta_k}(\mathbf{x}_k - \mathbf{W}_{1,k}(\mathbf{A}\mathbf{x}_k - \mathbf{b})).$$

Further analysis

Recall LISTA-CP model:

$$\mathbf{x}_{k+1} = \eta_{\theta_k}(\mathbf{x}_k - \mathbf{W}_{1,k}(\mathbf{A}\mathbf{x}_k - \mathbf{b})).$$

Assume $\mathbf{b} = \mathbf{A}\mathbf{x}_* +$ noise, where $\mathrm{supp}(\mathbf{x}_*)$ is uniformly distributed.

Further analysis

Recall LISTA-CP model:

$$\mathbf{x}_{k+1} = \eta_{\theta_k}(\mathbf{x}_k - \mathbf{W}_{1,k}(\mathbf{A}\mathbf{x}_k - \mathbf{b})).$$

Assume $\mathbf{b} = \mathbf{A}\mathbf{x}_* +$ noise, where $\mathrm{supp}(\mathbf{x}_*)$ is uniformly distributed.

Liu et al. [2019] shows that the recovery error and convergence rate only depend on

$$\sup_k \max_{1 \le i \ne j \le n} |\mathbf{w}_{i,k}^\top \mathbf{a}_j|$$

- $\mathbf{w}_{i,k}$ is the i-th column of $\mathbf{W}_{1,k}$; $\mathbf{a}_j$ is the j-th column of $\mathbf{A}$.
- $\mathbf{W}_{1,k}$ are scaled such that $\mathbf{w}_{i,k}^\top \mathbf{a}_i = 1$ for all $i = 1, 2, \cdots, n$.
- One might minimize the non-diagonal terms of $\mathbf{W}_{1,k}^\top \mathbf{A}$ independently for each k.
- An extension to *mutual coherence* in compressive sensing.

Parameter reduction: tie W_1 across iterations

Inspired by the analysis, let us try $\mathbf{W}_{1,k}$ tied for all k. Write it as $\mathbf{W}$.

- Tied LISTA (TiLISTA) iteration:

$$\mathbf{x}_{k+1} = \eta_{\theta_k}(\mathbf{x}_k - \gamma_k \mathbf{W}^\top(\mathbf{A}\mathbf{x}_k - \mathbf{b})).$$

Parameter reduction: tie W_1 across iterations

Inspired by the analysis, let us try $\mathbf{W}_{1,k}$ tied for all k. Write it as $\mathbf{W}$.

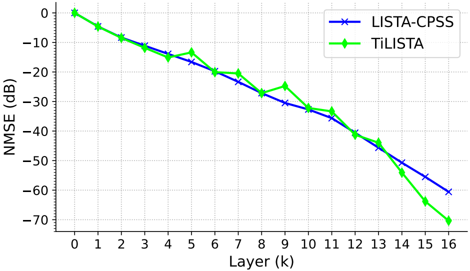
- Tied LISTA (TiLISTA) iteration:

$$\mathbf{x}_{k+1} = \eta_{\theta_k}(\mathbf{x}_k - \gamma_k \mathbf{W}^\top(\mathbf{A}\mathbf{x}_k - \mathbf{b})).$$

Parameters:

$$\mathcal{O}(mnK) \xrightarrow{\text{reduce}} \mathcal{O}(mn + K),$$

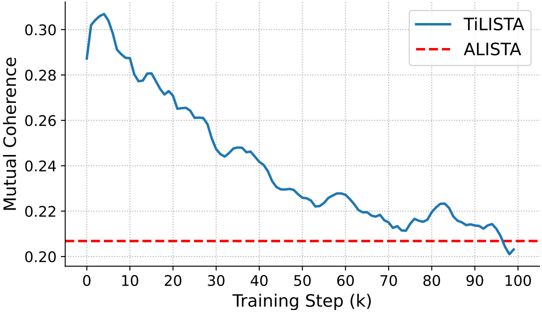
We learn only step sizes $\{\gamma_k\}_k$ and thresholds $\{\theta_k\}_k$ and a single matrix $\mathbf{W}$.



TiLISTA works even slightly better than LISTA-CPSS

Observation

We scale $\mathbf{W}$ such that $\mathbf{w}_i^\top \mathbf{a}_i = 1$ for $i = 1, \ldots, n$ and then measure $\max_{1 \leq i \neq j \leq n} |\mathbf{w}_i^\top \mathbf{a}_j|$ in TiLISTA. Compare it to ALISTA (next slide).



Good W needs to have small mutual coherence to A.

Analytic LISTA (ALISTA)

We use this principle to determine $\mathbf{W}$ *without training* [Liu et al., 2019] .

Analytic LISTA (ALISTA)

We use this principle to determine $\mathbf{W}$ *without training* [Liu et al., 2019] .

Two steps:

1. Compute approximately optimal $\tilde{\mathbf{W}}$:

$$\tilde{\mathbf{W}} \in \underset{\mathbf{W} \in \mathbb{R}^{m \times n}}{\operatorname{argmin}} \left\| \mathbf{W}^\top \mathbf{A} \right\|_F^2, \text{ s.t. } \mathbf{w}_i^\top \mathbf{a}_i = 1, \ \forall i = 1, 2, \cdots, n,$$

   which is a convex quadratic program (QP).

Analytic LISTA (ALISTA)

We use this principle to determine $\mathbf{W}$ *without training* [Liu et al., 2019] .

Two steps:

1. Compute approximately optimal $\tilde{\mathbf{W}}$:

   $$\tilde{\mathbf{W}} \in \underset{\mathbf{W} \in \mathbb{R}^{m \times n}}{\operatorname{argmin}} \left\| \mathbf{W}^\top \mathbf{A} \right\|_F^2, \text{ s.t. } \mathbf{w}_i^\top \mathbf{a}_i = 1, \ \forall i = 1, 2, \cdots, n,$$

   which is a convex quadratic program (QP).

2. With $\tilde{\mathbf{W}}$ fixed, learn $\{\gamma_k, \theta_k\}_k$ from data

Analytic LISTA (ALISTA)

We use this principle to determine $\mathbf{W}$ *without training* [Liu et al., 2019] .

Two steps:

1. Compute approximately optimal $\tilde{\mathbf{W}}$:

$$\tilde{\mathbf{W}} \in \underset{\mathbf{W} \in \mathbb{R}^{m \times n}}{\operatorname{argmin}} \left\| \mathbf{W}^\top \mathbf{A} \right\|_F^2, \text{ s.t. } \mathbf{w}_i^\top \mathbf{a}_i = 1, \ \forall i = 1, 2, \cdots, n,$$

   which is a convex quadratic program (QP).

2. With $\tilde{\mathbf{W}}$ fixed, learn $\{\gamma_k, \theta_k\}_k$ from data

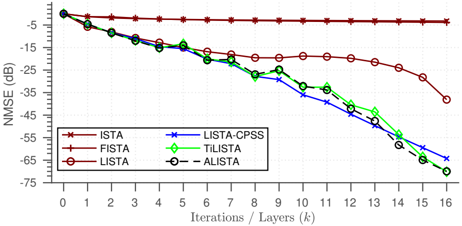
Parameters:

$$\mathcal{O}(mn + K) \stackrel{\text{reduce}}{\longrightarrow} \mathcal{O}(K).$$

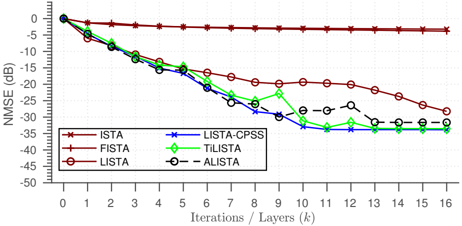
Training takes only minutes.

Numerical evaluation

Noiseless case
(SNR=∞)



Noisy case
(SNR=30dB)



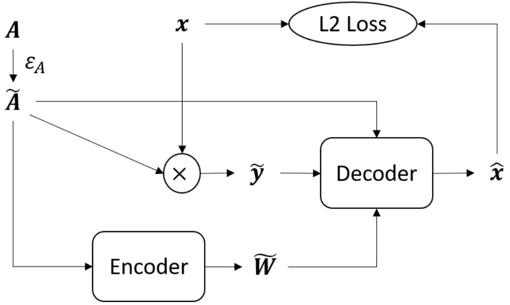
Robust ALISTA

Consider $\tilde{y} = \tilde{A}x + \varepsilon$ with $\tilde{A} = A + \varepsilon_A$. Given $\tilde{A}$ and $\tilde{y}$, recover x. Must handle varying $\tilde{A}$.

Unroll an algorithm into an NN to generate $\tilde{W}$ for $\tilde{A}$.

Method:

1. train an NN (called *encoder*) with many pairs of $(\tilde{A}, \tilde{W})$
2. train an ALISTA (called *decoder*) with many $(\tilde{A}, \tilde{y}, \tilde{W}, x)$
3. jointly train them with many $(\tilde{A}, \tilde{y}, \tilde{W}, x)$

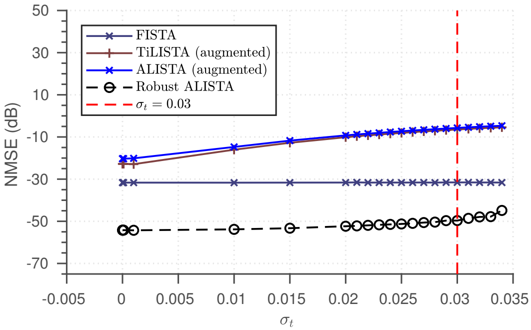


Numerical results

Fix an A. Training:

- Non-robust LISTA methods used their W matrices obtained with A.
- Robust ALISTA trained with perturbed A (Gaussian $\sigma = 0.03$).

Testing: All methods tested with perturbed A's (Gaussian $\sigma_1, \sigma_2, \cdots \leq 0.03$).



Robust ALISTA is significantly more robust.

HyperLISTA [Chen et al., 2021]

Introduce

- a hybrid-thresholding operator to bypass p^k largest entries and soft-threshold the rest
- analytic formulas for the parameters
- three hyper-parameters subject to grid search

Significance:

- allow the parameters to be "instance optimal"
- proves $\exists$ parameters to obtain *superlinear-like* error reduction

HyperLISTA learns $c_1, c_2, c_3 > 0$ and use them to set

$$\theta^k = c_1 \mu \Big\| A^{\dagger}(Ax^k - b) \Big\|_1, \qquad \text{soft threshold}$$

$$\beta^k = c_2 \mu \, \|x^k\|_0, \qquad \text{momentum stepsize}$$

$$p^k = c_3 \min\left(\log\left(\frac{\|A^{\dagger} b\|_1}{\|A^{\dagger}(Ax^k - b)\|_1} \right), n \right), \qquad \text{pass-through count}$$

The formulas are motivated by the analysis but use x^k instead of x^{true}.

HyperLISTA learns $c_1, c_2, c_3 > 0$ and use them to set

$$\theta^k = c_1 \mu \left\| A^\dagger (A x^k - b) \right\|_1, \qquad \text{soft threshold}$$

$$\beta^k = c_2 \mu \, \|x^k\|_0, \qquad \text{momentum stepsize}$$

$$p^k = c_3 \min \left(\log \left(\frac{\|A^\dagger b\|_1}{\|A^\dagger (A x^k - b)\|_1} \right), n \right), \qquad \text{pass-through count}$$

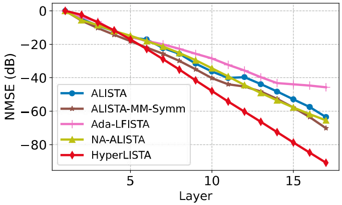
The formulas are motivated by the analysis but use x^k instead of x^{true}.

Parameters:

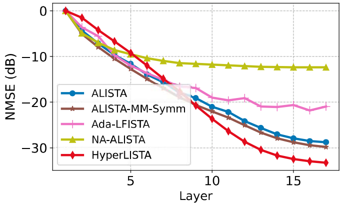
$$\mathcal{O}(K) \xrightarrow{\text{reduce}} 3.$$

Training can be done by grid search or a global optimization method.

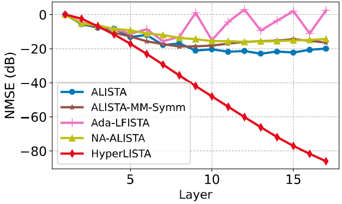
HyperLISTA is fast and robust



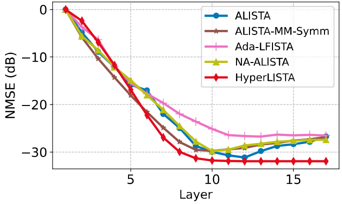
(a) Noiseless. No train/test mismatch.



(b) Sparsity ratio p changed to 0.15.



(c) Variance σ of non-zero elements changed to 2.



(d) Noise level changed to SNR=30dB.

Good analytic rules have better generalization perf.

LISTA [Gregor-LeCun, 2010] → $\left\{W_1^{(k)}, W_b^{(k)}, \alpha^{(k)}, \theta^{(k)}\right\}_{k=1}^{K}$ $\mathcal{O}(K(m+n)n)$

Eliminate W_1

LISTA-CPSS [Chen et al., 2018] → $\left\{W_b^{(k)}, \alpha^{(k)}, \theta^{(k)}\right\}_{k=1}^{K}$ $\mathcal{O}(Kmn)$

Eliminate W_b

ALISTA [Liu et al., 2019] → $\left\{\alpha^{(k)}, \theta^{(k)}\right\}_{k=1}^{K}$ $\mathcal{O}(K)$

Eliminate all parameters

HyperLISTA [Chen et al., 2021] → c_1, c_2, c_3 $\mathcal{O}(1)$

Training time: 10 hour → 6 mins

Uncovered LISTA topics

- [Moreau and Bruna, 2017] proposed to understand LISTA by the similarity between LISTA and a matrix-factorization method.
- [Xin et al., 2016] proposed learned iterative hard-thresholding-CP.
- [Wu et al., 2019] proposed gated mechanisms to improve LISTA.
- [Ito et al., 2019] proposed a minimum mean squared error (MMSE) estimator-based shrinkage function in LISTA.
- [Yang et al., 2020] proposed to use nonconvex-function-induced regularizers in LISTA.
- [Heaton et al., 2020] introduced a safeguard wrapper for LISTA methods applied to structured convex problems.
- When K is large or $K = \infty$, LISTA cannot be trained. Instead, we can use deep equilibrium[Bai et al., 2019, Winston and Kolter, 2020] and fixed-point network [Fung et al., 2022]. [Gilton et al., 2021] demonstrated better image recovery.

References:

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016.

Tianlong Chen, Weiyi Zhang, Zhou Jingyang, Shiyu Chang, Sijia Liu, Lisa Amini, and Zhangyang Wang. Training stronger baselines for learning to optimize. *Advances in Neural Information Processing Systems*, 33:7332–7343, 2020.

Xiaohan Chen, Jialin Liu, Zhangyang Wang, and Wotao Yin. Hyperparameter tuning is all you need for lista. *Advances in Neural Information Processing Systems*, 34: 11678–11689, 2021.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th international conference on international conference on machine learning*, pages 399–406, 2010.

James Harrison, Luke Metz, and Jascha Sohl-Dickstein. A closer look at learned optimization: Stability, robustness, and inductive biases. *arXiv preprint arXiv:2209.11208*, 2022.

Jialin Liu, Xiaohan Chen, Zhangyang Wang, and Wotao Yin. Alista: Analytic weights are as good as learned weights in lista. In *International Conference on Learning Representations (ICLR)*, 2019.

Luke Metz, Niru Maheswaranathan, Jeremy Nixon, Daniel Freeman, and Jascha Sohl-Dickstein. Understanding and correcting pathologies in the training of learned optimizers. In *International Conference on Machine Learning*, pages 4556–4565. PMLR, 2019.

Luke Metz, C Daniel Freeman, James Harrison, Niru Maheswaranathan, and Jascha Sohl-Dickstein. Practical tradeoffs between memory, compute, and performance in learned optimizers. In *Conference on Lifelong Learning Agents*, pages 142–164. PMLR, 2022.

Olga Wichrowska, Niru Maheswaranathan, Matthew W Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Nando Freitas, and Jascha Sohl-Dickstein. Learned optimizers that scale and generalize. In *International Conference on Machine Learning*, pages 3751–3760. PMLR, 2017.

Yuhuai Wu, Mengye Ren, Renjie Liao, and Roger Grosse. Understanding short-horizon bias in stochastic meta-optimization. *arXiv preprint arXiv:1803.02021*, 2018.

Yuanhao Xiong, Li-Cheng Lan, Xiangning Chen, Ruochen Wang, and Cho-Jui Hsieh. Learning to schedule learning rate with graph neural networks. In *International Conference on Learning Representation (ICLR)*, 2022.