Learning to Optimize

Wotao Yin

UCLA Math and Alibaba US Damo Academy

East Coast Optimization Meeting 2021

Co-authors, paper, and code

- Tianlong Chen, Xiaohan Chen, Wuyang Chen, Zhangyang Wang (UT Austin)
- Howard Heaton (UCLA)
- Jialin Liu (Alibaba US)
- Learning to Optimize: A Primer and A Benchmark. arXiv:2103.12828
- Code: https://github.com/VITA-Group/Open-L2O

1.

Why and how to learn to optimize

Machine learning vs Optimization

Answers are given as existing data or experience

ML learns from data and experience to give answers in the future

Induction

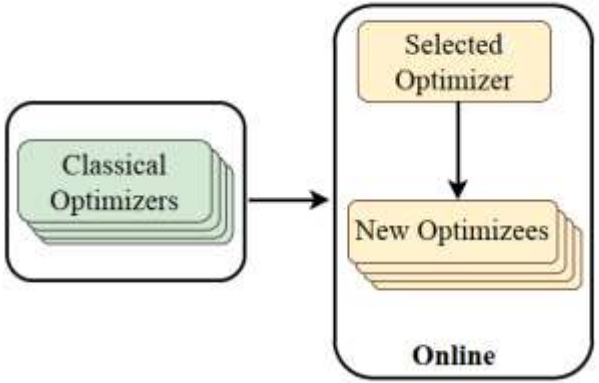
No answer is given; but we know how to evaluate how answers are.

OPT will find answers with best evaluations

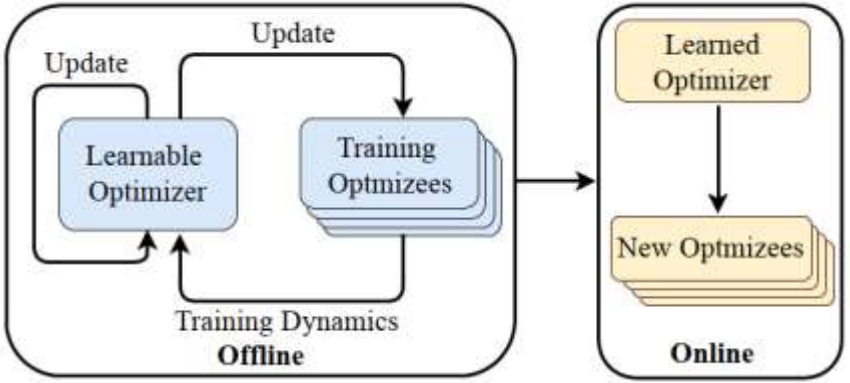
Prescription

L2O uses past optimization experience (i.e., observed performance) to “optimize better in the future.”

Classic optimization



Learning-to-optimize



Classic vs Learning-to-optimize (L2O)

Classic optimization methods

Methods are typically hand-built from basic components – GD, CG, Newton steps, LS, stochastic sampling, and so on – in a theoretically justified manner.

Most are written in a few lines.

Many come with theories, performance guarantees, and interpretations.

To solve a problem:

- Recognize its type
- Select an existing method, tune a few parameters

Learning-to-optimize methods

Methods are developed by training

May lack theory, be difficult to interpret, but performance improved during training

Can borrow ideas from classic optimization

Training takes time; applying is fast

Can be applied to

- Improve solution speed, but not quality
- Improve solution quality, not nec. speed

When shall we consider L2O

- Having **examples** of good solutions
- But it is hard to formularize them analytically (e.g., inverse problems)
- Or an accurate formulation is too difficult to solve

L2O can help find better solutions!

- Solving similar optimization problems **repeatedly**
- The task distribution is concentrated and can be represented by examples

L2O can help find a “fast shortcut” to the solutions!

Basic formulation

Consider $\min_{\mathbf{x}} f(\mathbf{x})$

- GD iteration: $\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha \nabla f(\mathbf{x}_t)$
- L2O tends to free up parameters and use more information
  - Let $\mathbf{z}_t$ include all iterates and gradients so far
  - Use $\mathbf{x}_{t+1} = \mathbf{x}_t - g(\mathbf{z}_t, \phi)$ where g is parameterized by ϕ
  - An example L2O formulation (Andrychowicz et al'NIPS16):

$$\min_{\phi} \mathbb{E}_{f \in \mathcal{T}} \left[\sum_{t=1}^{T} w_t f(\mathbf{x}_t) \right]$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t - g(\mathbf{z}_t, \phi), \ t = 1, \ldots, T-1$$

model-based vs model-free

- g has a form of an existing method or uses it as a starting point
- L2O searches for the **best values of some parameters**
- You may combine this L2O with classic methods in various ways

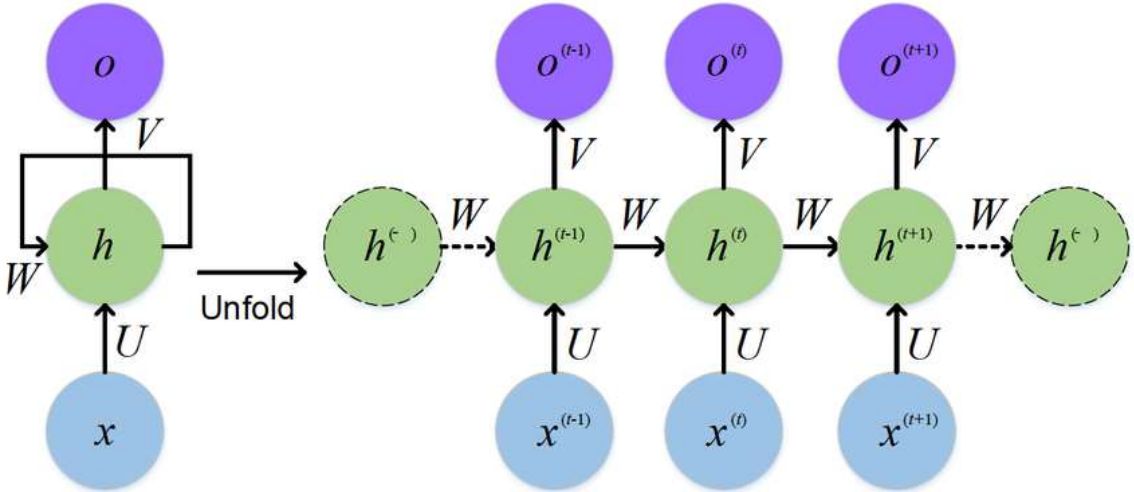
- g is based on universal approximators, e.g., multi-layer neural networks or recurrent neural networks.
- L2O is set to discover **completely new** update rules without referring to any existing updates (other than being iterative)

2.

Model-Free L2O

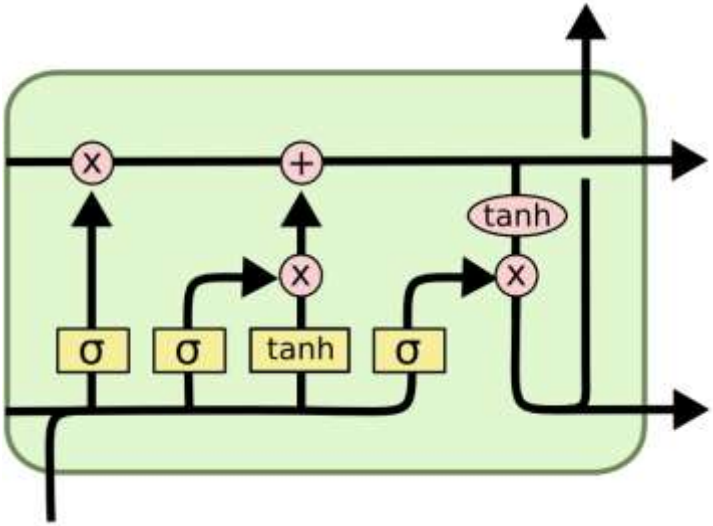
based on RNNs, especially LSTMs

RNN and unfolding



Wichrowska et al'17; Metz et al'ICML19; Li-Malik'ICLR17; Bello et al'ICC17; Jiang et al'18;

Many model-free L2O uses LSTM



Andrychowicz et al'NIPS16; Chen et al'ICML17; Lv-Jiang-Li'17; Cao et al'NeurIPS19; Xiong-Hsieh'20

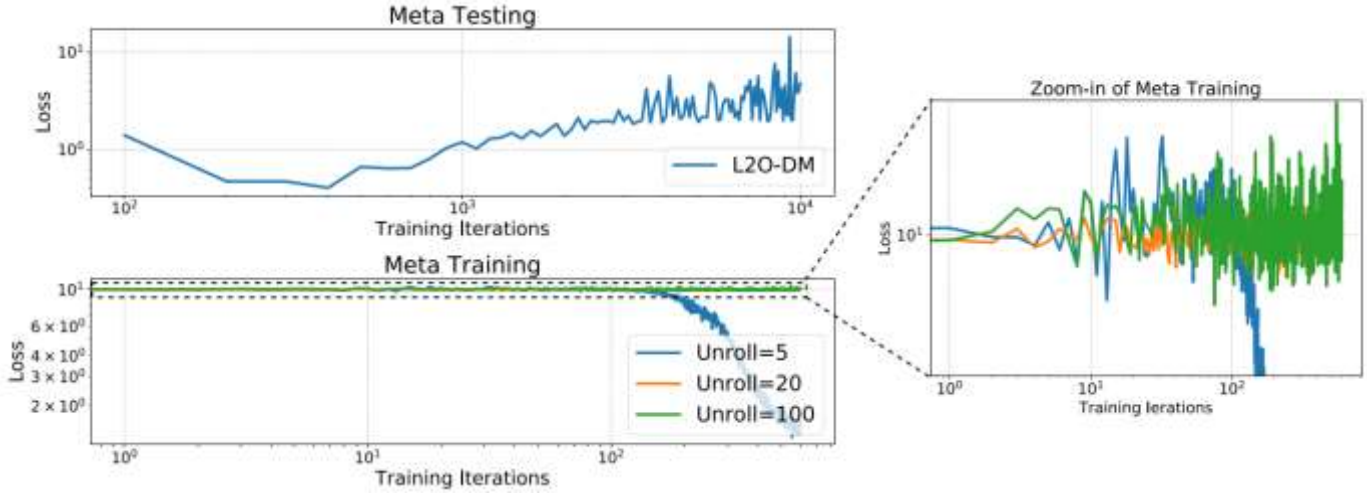
Optimizer Architecture	Input Feature	Meta Training Objective	Additional Technique	Evaluation Metric
LSTM	Gradient	Meta Loss	Transform input gradient ∇ into $\log(\nabla)$ and $\text{sign}(\nabla)$	Training Loss
LSTM	Objective Value	Objective Value	N/A	Objective Value
LSTM	Gradient	Meta Loss	Random Scaling Combination with Convex Functions	Training Loss
Hierarchical RNNs	Scaled averaged gradients, relative log gradient magnitudes, relative log learning rate	Log Meta Loss	Gradient History Attention Nesterov Momentum	Training Loss
MLP	Gradient	Meta Loss	Unbiased Gradient Estimators	Training Loss Testing Loss
RNN Controller	Loss, Gradient	Meta Loss	Coordinate Groups	Training Loss
Searched Mathematical Rule by Primitive Functions	Scaled averaged gradients	Meta Loss	N/A	Testing Accuracy
Multiple LSTMs	Gradient, momentum, particle's velocity and attraction	Meta Loss and Entropy Regularizer	Sample- and Feature- Attention	Training Loss
RNN	Input Images, Input Gradient	Meta Loss	N/A	Standard and Robust Test Accuracies
LSTM	Input Gradient	Meta Loss	N/A	Training Loss and Robust Test Accuracy

Meta learning

- Rough definition: the approach that uses a method to improve learning algorithm(s), a.k.a. “learning to learn”
- Training those algorithms is called “meta training”
- Testing those algorithms is called “meta testing”
- L2O is not entirely meta learning, and vice versa

Challenges: unroll lengths

- Long unroll causes high memory cost
- Short unroll fails to generalize to more iterations



Reinforcement learning (RL)

- (Li-Malik'ICLR17) Learns a policy to determine the update rule; uses training loss reduction as reward
- (Bello et al'ICC17) uses RL to select a sequence of symbolic elements

3.
Model-Based L2O

Plug-n-Play, Unrolling, Safeguarding, etc.

Plug-and-Play

- Replace a part of a classic method by learned operator
- (Venkatakrishnan et al'GlobalSIP13) PnP ADMM:

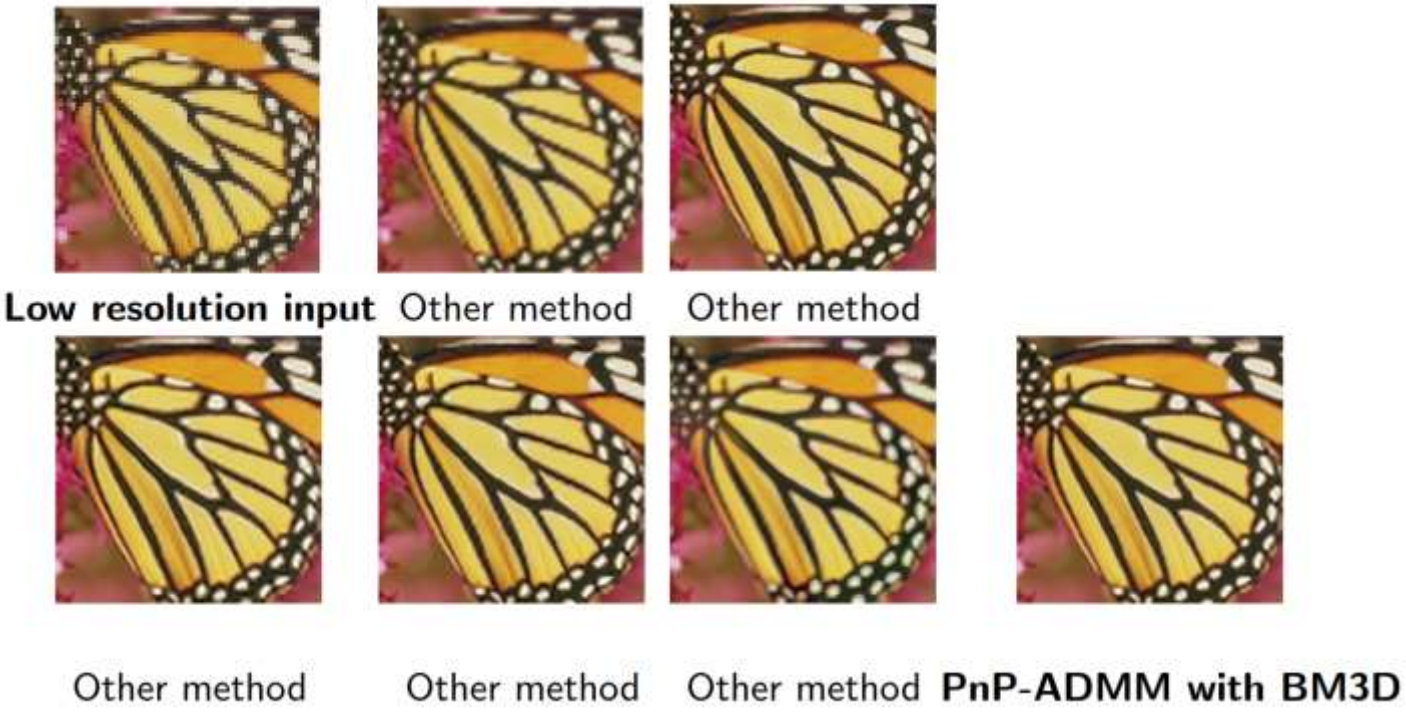
$$\underset{x\in\mathbb{R}^d}{\text{minimize}} \quad f(x)+\gamma g(x)$$

$$\begin{aligned} x^{k+1} &= \text{Prox}_{\sigma^2 g}(y^k - u^k) \\ y^{k+1} &= \text{Prox}_{\alpha f}(x^{k+1} + u^k) \\ u^{k+1} &= u^k + x^{k+1} - y^{k+1}. \end{aligned}$$

$$\begin{aligned} x^{k+1} &= H_\sigma(y^k - u^k) \\ y^{k+1} &= \text{Prox}_{\alpha f}(x^{k+1} + u^k) \\ u^{k+1} &= u^k + x^{k+1} - y^{k+1}. \end{aligned}$$

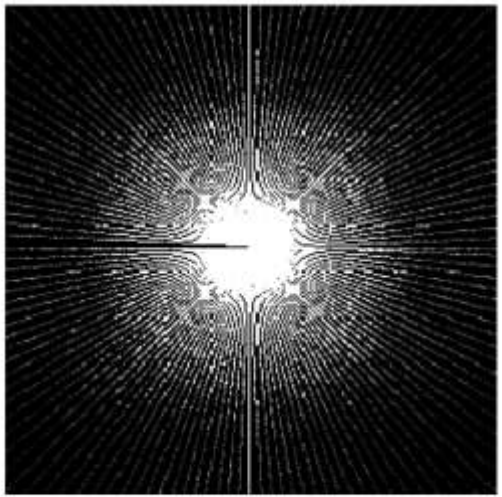
- (Ryu et al'ICML19) guarantees convergence by combining Lipschitz and contraction properties

Example: Super resolution

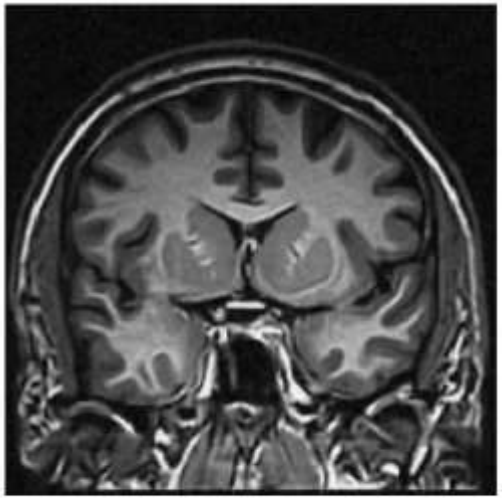


(Chen-Wang-Elgendy'17)

Example: Compressed-sensing MRI



Radial sampling k-space



Recovery 19.09dB

(Ryu et al'ICML19)

PSNR (in dB) for 30% sampling with additive Gaussian noise $\sigma_e = 15$. RealSN generally improves the performance.

Sampling approach		Random		Radial		Cartesian	
Image		Brain	Bust	Brain	Bust	Brain	Bust
Zero-filling		9.58	7.00	9.29	6.19	8.65	6.01
TV		16.92	15.31	15.61	14.22	12.77	11.72
RecRF		16.98	15.37	16.04	14.65	12.78	11.75
BM3D-MRI		17.31	13.90	16.95	13.72	14.43	12.35
PnP-FBS	BM3D	19.09	16.36	18.10	15.67	14.37	12.99
	DnCNN	19.59	16.49	18.92	15.99	14.76	14.09
	RealSN-DnCNN	**19.82**	**16.60**	**18.96**	**16.09**	**14.82**	**14.25**
	SimpleCNN	15.58	12.19	15.06	12.02	12.78	10.80
	RealSN-SimpleCNN	17.65	14.98	16.52	14.26	13.02	11.49
PnP-ADMM	BM3D	19.61	**17.23**	18.94	**16.70**	14.91	13.98
	DnCNN	19.86	17.05	19.00	16.64	14.86	14.14
	RealSN-DnCNN	**19.91**	17.09	**19.08**	16.68	**15.11**	**14.16**
	SimpleCNN	16.68	12.56	16.83	13.47	13.03	11.17
	RealSN-SimpleCNN	17.77	14.89	17.00	14.47	12.73	11.88

(Ryu et al'ICML19)

Unrolling example: LASSO/ISTA

LASSO model:

$$x^{\text{lasso}} \leftarrow \underset{x}{\text{minimize}} \ \frac{1}{2}\|b - Ax\|_2^2 + \lambda\|x\|_1$$

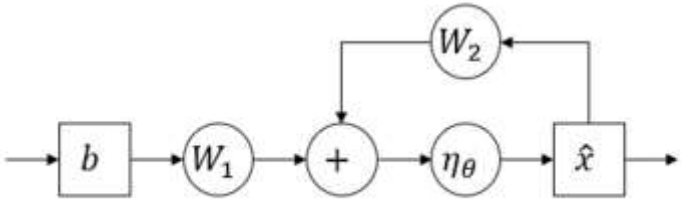
Iterative Shrinkage and Thresholding Algorithm (**ISTA**):

$$x^{(k+1)} = \eta_{\frac{\lambda}{L}}\left(x^{(k)} + \frac{1}{L}A^T(b - Ax^{(k)})\right)$$

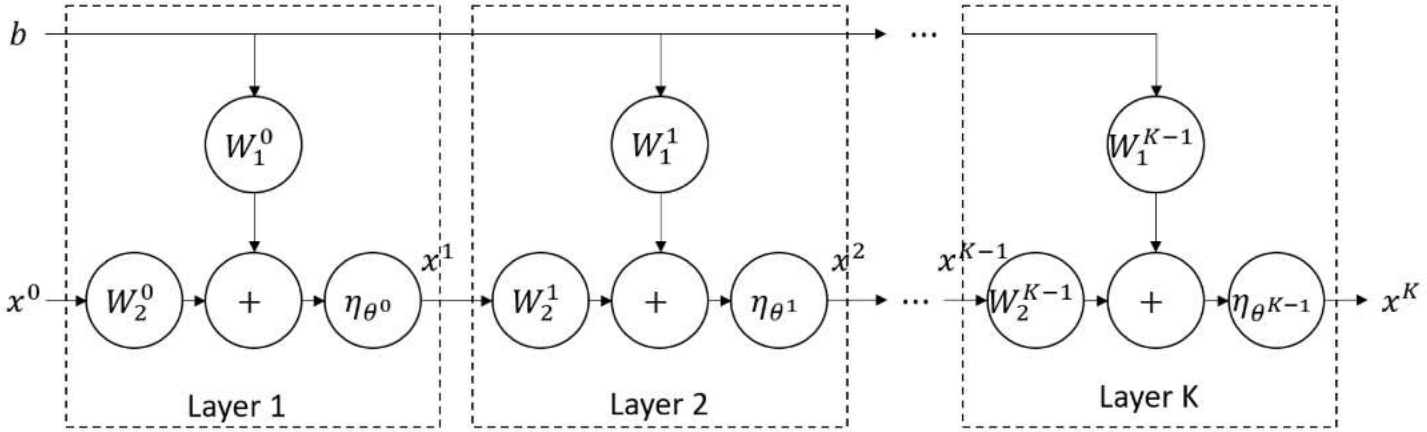
Rewrite ISTA as

$$x^{(k+1)} = \eta_\theta(W_1 b + W_2 x^{(k)}),$$

where $W_1 = \frac{1}{L}A^T, W_2 = I_n - \frac{1}{L}A^T A$ and $\theta = \frac{\lambda}{L}$.



Unrolling



- Truncate an iterative algorithm, and view as a feedforward neural network that can be end-to-end trained
- Often no need to learn all parameters (Chen et. al. NeurIPS'18 & Liu et. al. ICLR'19)
- Popular and successful in inverse problems, PDEs, and graphical models

Challenges

Practice: specific design tied to the problem

Theory: although “model-based”, little theory exists due to black-box training

- Capacity: is L2O provably good/better?
- Trainability: how to train, any guarantee?
- Generalization: when and what if L2O fails?
- Interpretability: what L2Os have learned?

Safe-guarding

(Heaton et al.20) L2O convergence can be ensured by incorporating an "energy" *E*, which is related to nonmonotone line search.

$$x^{k+1} = \begin{cases} \text{L2O update } z^k & \text{if } E^k(z^k) \le E^k(x^k) \\ \text{classic update } T(x^k) & \text{otherwise} \end{cases}$$

When L2O fails to decrease the energy, the classic update T will take over in that iteration

4.
Benchmarks

A few representative test problems

- Convex sparse recovery/optimization
- Nonconvex Rastrigin function minimization
- Training neural networks

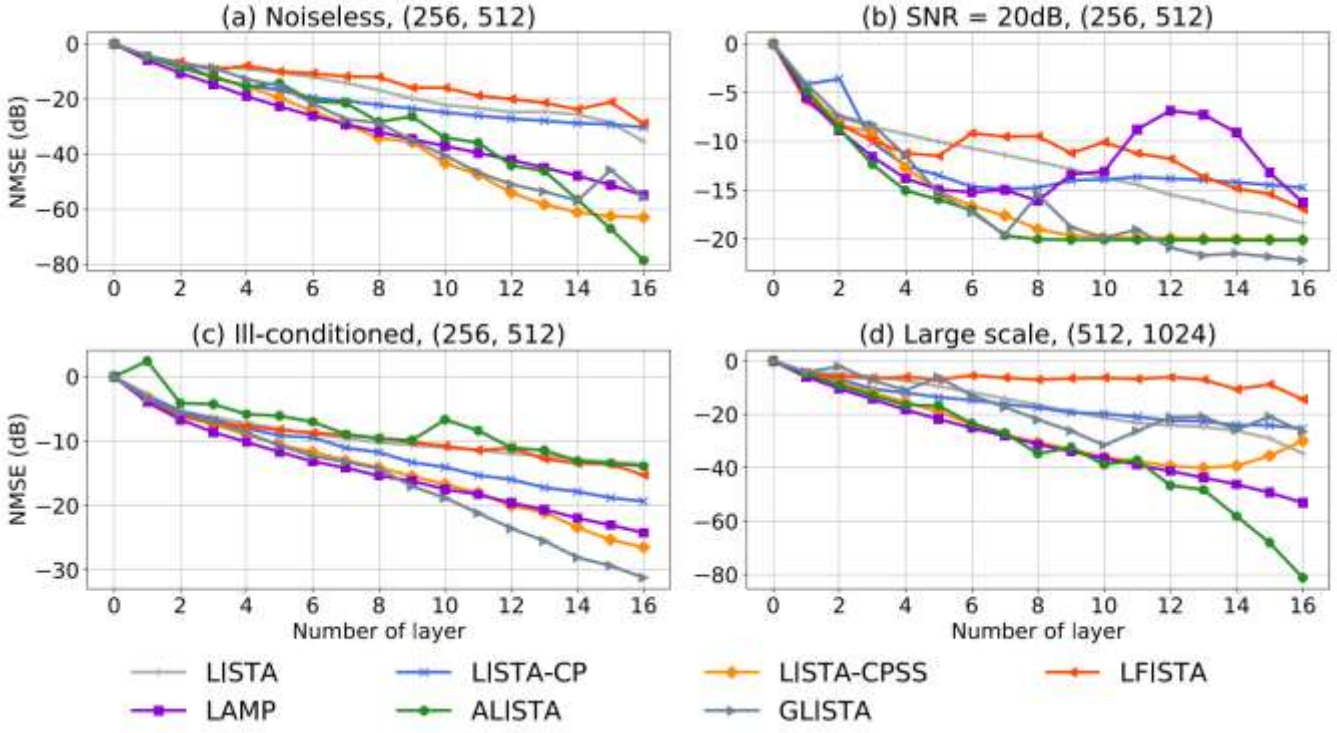
Sparse optimization

- Recover a sparse vector from noisy measurements

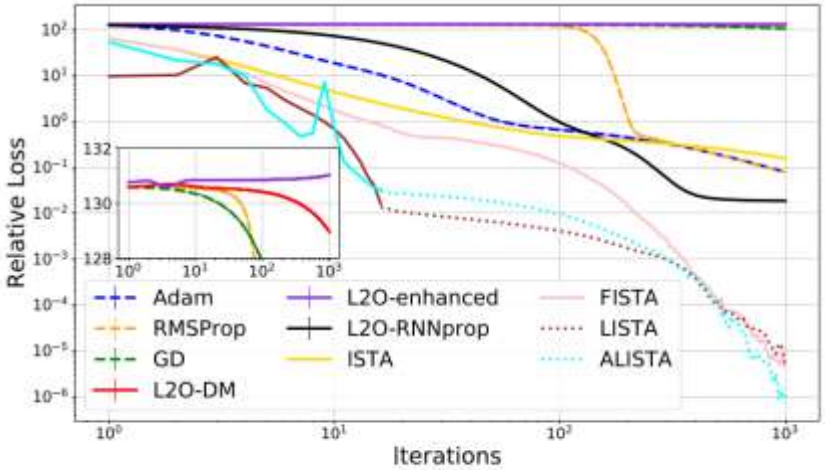
$$b_q = Ax_q^* + \varepsilon_q$$

- A Is fixed, but sparse vectors
- Without noise, we expect exact recovery
- With noise present, we expect approx. recovery
- Training loss is squared-L2 to the true signal

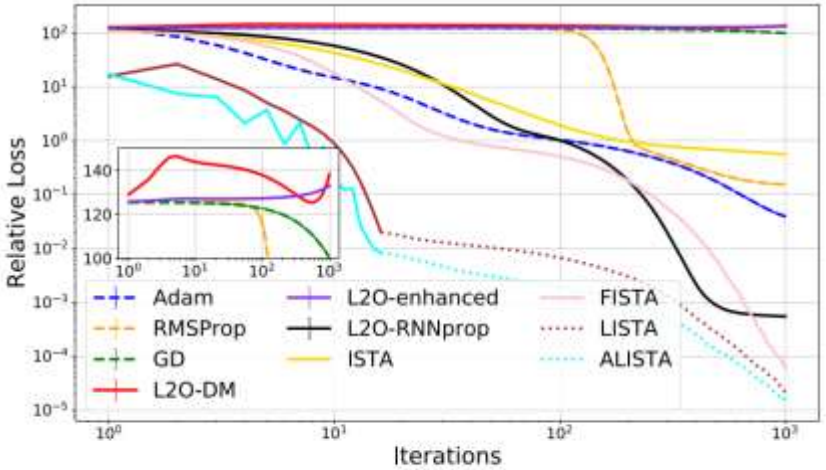
Learn to recover



Learn to solve the LASSO model



(a) $(m,n)=(5,10)$



(b) $(m,n)=(25,50)$

Rastrigin function

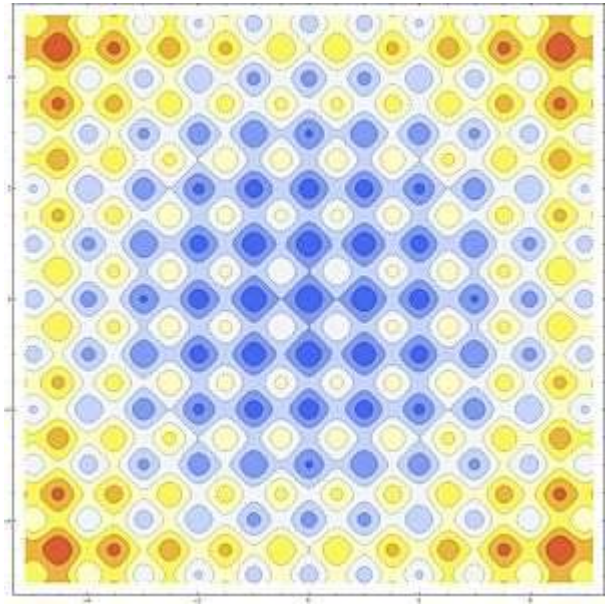
- A popular non-convex test function

$$f(\boldsymbol{x}) = \frac{1}{2}\sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} \alpha \cos\left(2\pi x_i\right) + \alpha n$$

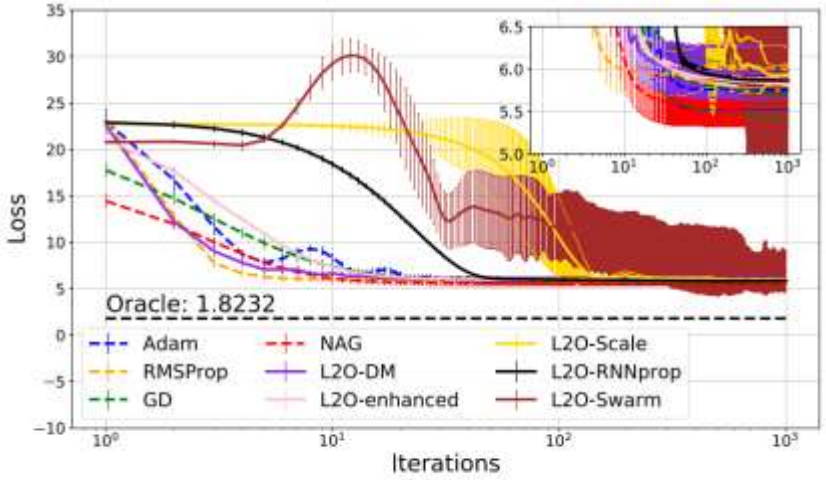
- We generalized it to higher dimenions

$$f_q(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}_q\boldsymbol{x} - \boldsymbol{b}_q\|_2^2 - \alpha \boldsymbol{c}_q \cos(2\pi \boldsymbol{x}) + \alpha n$$

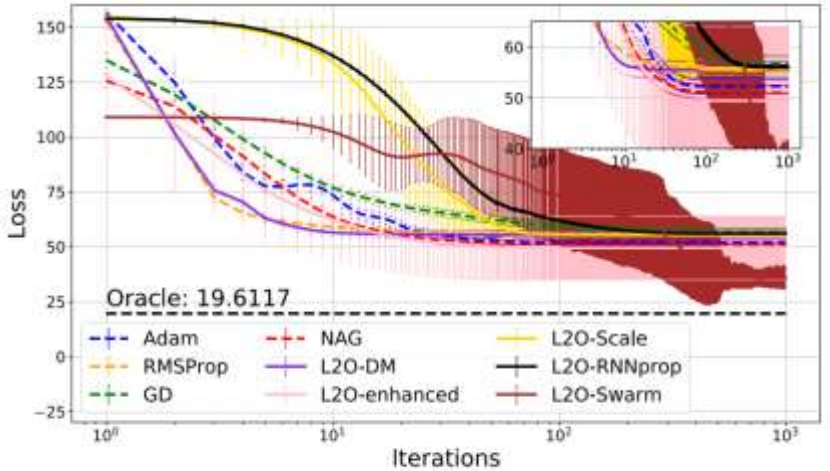
- Data are sampled from Gaussian



Speed and solution quality



(a) n=2

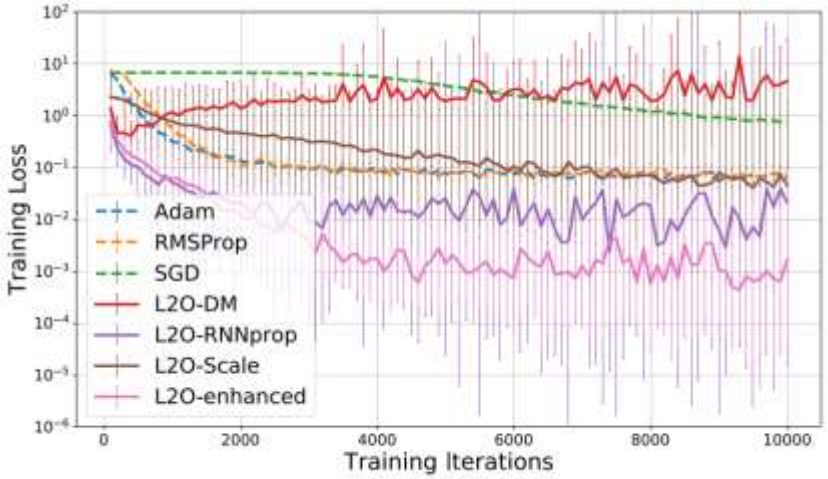


(b) n=10

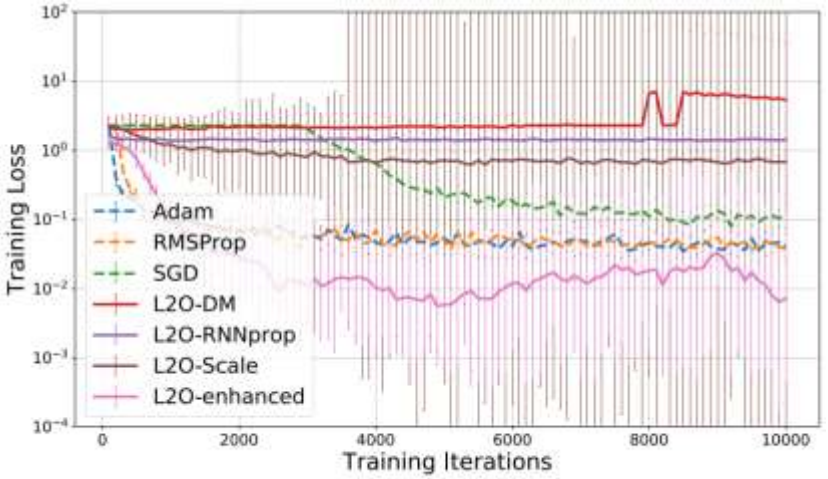
Neural network training

- Can L2O outperform analytic ones?
- Generalization to unseen architectures and data?
- In-distribution training: MLP with a 20-dim hidden layer and sigmoid activation; MNIST; minimize cross-entropy loss; random initial data
- Out-of-distribution testing:
  - ReLU instead of Sigmoid
  - Another ConvNet on the MNIST

Testing: training performance



(a) MLP



(b) ConvNet

5.
Uncovered topics

- Progress of solving MIPs, SATs, etc.
- Unrolling second-order (e.g., quasi-Newton) methods
- Learn to improve the model or input parameters
- Use a classic solver, possibly with parameters, as a layer in a large network
- ……

6.
Open questions

- Lack of training data due to privacy or proprietary protections
- Unbalanced training data
- Un-safeguarded L2O methods may fail, unsuitable for critical scenarios
- Difficult to interpret, cannot do “what-if” analysis

Thanks!

Any questions?

You can email us at:

Zhangyang Wang <atlaswang@utexas.edu>

Wotao Yin <wotao.yin@alibaba-inc.com>

Credits

Special thanks to all the students / collaborators who worked with us and released the code:

- Tianlong Chen, Xiaohan Chen, Wuyang Chen
- Howard Heaton
- Jialin Liu