Explainable AI via Learning to Optimize

Howard Heaton[1] + Samy Wu Fung[2]

[1]Typal Academy, [2]Colorado School of Mines

Talk Overview

Setting

Problems where approximate optimization models can be hand-crafted

Learning to Optimize (L2O)

Make parameterized optimization model and use training data to tune it, *i.e.*

$$\text{(model output)} \triangleq \text{argmin (prior knowledge)} + \text{(data-driven terms)}$$

Contribution

- Demo how to build intuitive L2O models
- Provide certificates to explain whether model inferences are trustworthy

Historic Comparison of Two Approaches

Machine Learning

$$N_\Theta(d) = \sigma(W^m \cdot + b^m) \circ \cdots \circ \sigma(W^1 d + b^1)$$

- ✔ adapt to available data
- ✖ satisfy constraints / optimality
- ✔ expressive capacity
- ✔ flexible architectures

Traditional Optimization

$$\operatorname*{argmin}_{x \in C} f(x)$$

- ✖ adapt to available data
- ✔ guaranteed optimality
- ✔ interpretable models
- ✔ scalable first-order algorithms

L2O Overview

- Originated with LISTA[1] where authors took existing algorithm (ISTA) and replaced analytic terms for affine mappings with parameters and tuned them with training data to quickly solving sparse coding problems
- Inspired by optimization (may be written via fixed points)
- Can be used in embedded form (*e.g.* optimization is one layer in model)
- Has switched emphasis (in our work) to using *many* iterations

[1] Gregor and LeCun. *Learning fast approximations of sparse coding*. 2010.

Related Work

- **Plug and Play** (parameters not tuned on data used for inferences)

  Plug externally trained model in an algorithm as a proximal/gradient update

- **Deep Unfolding** (does not run to convergence, limited guarantees)

  Apply “small” # of updates, with (possibly) different parameters in each step

- **Predict-then-Optimize** (single “layer” usage, special case of L2O)

  Learn mapping from data to apt optimization problem, and then solve

What is Explainable?

Model Design $\rightarrow$ Inference Properties[2]

A model is explainable provided a domain expert can identify the core design elements of a model and how they translate to expected inference properties

Inference Properties $\rightarrow$ Model Design + Training Data

An inference is explainable provided its properties can be linked to the model's design and intended use, enabling identification of trustworthy inferences

Explainable models *and* inferences are achieved via L2O with our certificates[3]

[2]These are properties for inferences on data matching the distribution of training data

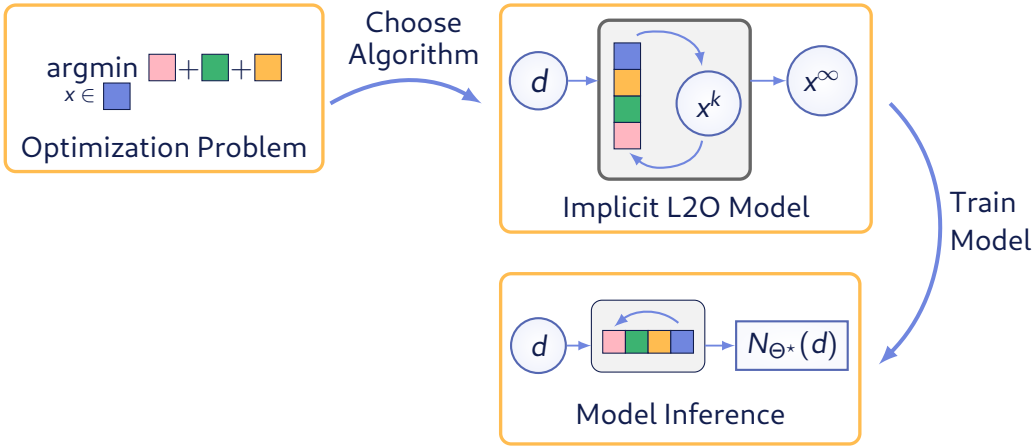
[3]These certificates can be used for post-conditions in production code

Outline

1. How to Build an Explainable L2O Model

2. Trustworthiness Certificates

L2O via Building Blocks



Key L2O Steps

1. Make a model by parameterizing an optimization problem via Θ to get

$$N_\Theta(d) = S_\Theta(x_{\Theta,d}), \quad \text{where} \quad x_{\Theta,d} \triangleq \underset{x}{\operatorname{argmin}}\, f_\Theta(x;\ d)$$

Note: We focus on case where S_Θ is identity, but this part can take many forms (*e.g.* S_Θ can be a classifier)

Note: Constraints can be included in this formulation (via indicator functions)

2. Forward prop by applying an apt first-order algorithm until convergence

Note: For best performance, test like you train (*i.e.* use same # of iterations)

3. Backprop consists of using built-in autograd on *last step of forward prop*

Toy Example – Signal Recovery via Dictionary Learning

- **Task**

  Recover a signal $x_d^\star$ from linear measurements $d = Ax_d^\star$

- **Key Knowledge**

  Signal $x_d^\star$ has low dimensional structure (but is *not* sparse)

- **L2O Model**

  For a "sparsifying matrix" K, we estimate

$$x_d^\star \approx \underset{x}{\operatorname{argmin}} \|Kx\|_1 \quad \text{s.t.} \quad Ax = d$$

Toy Example – Signal Recovery via Dictionary Learning

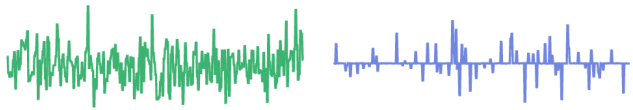


Figure 1: Applying the learned K sparsifies $x_d^\star$ (shown for test data d)

Fix weights $\Theta = K \in \mathbb{R}^{250\times 250}$, noting $x_d^\star \in \mathbb{R}^{250}$ and $d \in \mathbb{R}^{100}$, and set

$$N_\Theta(d) \triangleq \operatorname*{argmin}_x \|Kx\|_1 \text{ s.t. } Ax = d$$

For a distribution of measurement/signal pairs $(d, x_d^\star)$, train by minimizing

$$\min_\Theta \mathbb{E}_d \left[\|x_d^\star - N_\Theta(d)\|^2 \right]$$

(*More details for this example are in later slide*)

```
x_fxd_pt = find_fixed_point(d)
y = apply_opt_update(x_fxd_pt, d)
loss = criterion(y, labels)
loss.backward()
optimizer.step()
```

Figure 2: Sample PyTorch code for backpropagation. The `find_fixed_point` function repeatedly applies `apply_opt_update` until a fixed point is (approximately) found.

(Informal) Theorem[4]

Backpropping through the final step of a fixed point algorithm (as shown above) yields a *preconditioned* gradient

[4]Wu Fung, *et al. JFB: Jacobian-Free Backpropagation for Implicit Networks*, 2022.

Outline

1. How to Build an Explainable L2O Model

2. Trustworthiness Certificates

What Inferences are Trustworthy?

An inference is trustworthy provided its properties can satisfactorily be linked to a model's design and intended use

We make this concrete using certificates

- Each property in model design corresponds to a certificate for inferences
- Each certificate is a tuple: (property name, label)
- Labels can be "pass," "warning," or "fail"
- (All certificate labels read "pass") $\Longrightarrow$ trustworthy inference

How to Compute Certificate Labels

Labels are derived from nonnegative inference property value (smaller is better):

$$(\text{inference}) \to (\text{property value}) \to (\text{label})$$

Set p_p to desired probability for "pass" labels[5] (similarly for p_w and warnings) and

$$\text{label}(\alpha) \triangleq \begin{cases} \text{pass} & \text{if } \alpha \in [0, c_p] \\ \text{warning} & \text{if } \alpha \in (c_p, c_w] \\ \text{fail} & \text{otherwise} \end{cases}$$

with c_p such that $\mathbb{P}_{d \sim \mathcal{D}}\big[(\text{property value})(N_{\Theta^\star}(d)) \leq c_p\big] = p_p$, and similarly for c_w

[5] Here $p_p = 0.95$ means 95% of inferences $N_\Theta(d)$ pass with d drawn from training distribution $\mathcal{D}$

Property Value Functions

Concept	Quantity	Formula
Sparsity	# Nonzeros	$\|x\|_0$
Measurements	Relative Error	$\|Ax - d\|/\|d\|$
Constraints	Distance to Set	$d_C(x)$
Regularization	Proximal Residual	$\|x - \text{prox}_{f_\Theta}(x)\|$

Table 1: Example formulas for property value functions.

Toy Example (Continued)

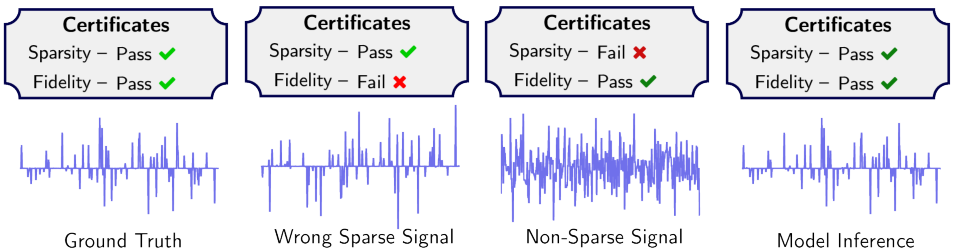
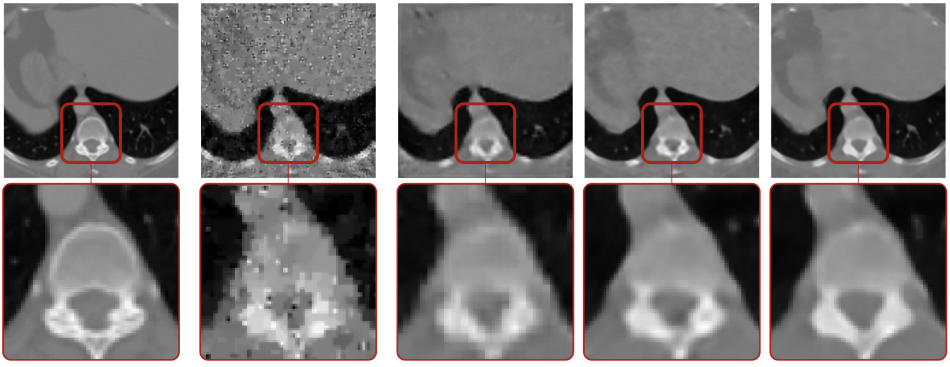


Figure 3: For sample d from test data, sparsified $KN_\Theta(d)$ of each inference $N_\Theta(d)$ is shown

CT Image Reconstruction



Ground truth	TV Min	U-Net	F-FPN	Implicit L2O
SSIM: 1.000	SSIM: 0.379	SSIM: 0.806	SSIM: 0.853	SSIM: 0.891
PSNR: ∞	PSNR: 21.17	PSNR: 27.83	PSNR: 30.32	PSNR: 31.87

Figure 4: Comparison of techniques, ranging from analytic to fully data-driven

$$(\text{L2O Model}) = N_\Theta(d) \triangleq \underset{x\in[0,1]^n}{\operatorname{argmin}} f_\Theta(Kx) \;\text{ s.t. }\; \|Ax - d\| \leq \delta$$

Summary

- Models with optimization layers that have tunable parameters can be readily designed and explained by domain experts
- With a well-chosen algorithm, implicit L2O models can be trained using JFB
- Certificates can be used to identify whether properties of each inference are consistent with training data (*i.e.* via post-conditions in software)

docs site	xai-l2o.research.typal.academy
preprint	arXiv.org/abs/2204.14174
reprint	Nature Scientific Reports (*accepted*, *coming soon*)