Bilevel Learning for Inverse Problems

Matthias J. Ehrhardt

Department of Mathematical Sciences, University of Bath, UK

September 15, 2021

Joint work with:
L. Roberts (ANU, Australia)

F. Sherry, M. Graves, G. Maierhofer, G. Williams, C.-B. Schönlieb (all Cambridge, UK), M. Benning (Queen Mary, UK), J.C. De los Reyes (EPN, Ecuador)

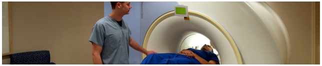
The Leverhulme Trust

UKRI Engineering and Physical Sciences Research Council

THE FARADAY INSTITUTION

Outline

1) Motivation



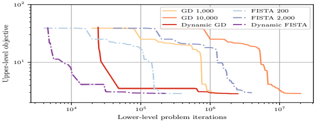
$$\min_x \frac{1}{2}\|Ax - y\|_2^2 + \lambda\mathcal{R}(x)$$

2) Bilevel Learning

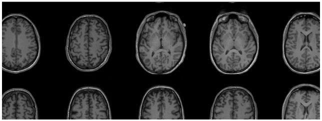
$$\min_{x,y} f(x, y)$$

$$x \in \arg\min_z g(z, y)$$

3) Efficient solution?
Yes, e.g. inexact DFO algorithms
Ehrhardt and Roberts JMIV 2021



4) High-dimensional learning?
Yes, e.g. learn MRI sampling
Sherry et al. IEEE TMI 2020



Inverse problems

$$Ax = y$$

x : desired solution

y : observed data

A : mathematical model

Goal: recover x given y

Inverse problems

$$Ax = y$$

x : desired solution

y : observed data

A : mathematical model

Goal: recover x given y

Hadamard (1902): We call an inverse problem $Ax = y$ **well-posed** if

(1) a solution x^* **exists**

(2) the solution x^* is **unique**

(3) x^* depends **continuously** on data y.

Otherwise, it is called **ill-posed**.

Jacques Hadamard

Most interesting problems are **ill-posed**.

How to solve inverse problems?

Variational regularization ($\sim$1990)
Approximate a solution x^* of $Ax = y$ via

$$\hat{x} \in \arg\min_x \left\{ \mathcal{D}(Ax, y) + \lambda \mathcal{R}(x) \right\}$$

- $\mathcal{D}$ data fidelity, related to noise statistics
- $\mathcal{R}$ **regularizer**: penalizes unwanted features, ensures stability and uniqueness
- λ **regularization parameter**: $\lambda \geq 0$. If $\lambda = 0$, then an original solution is recovered. As $\lambda \to \infty$, more and more weight is given to the regularizer $\mathcal{R}$.

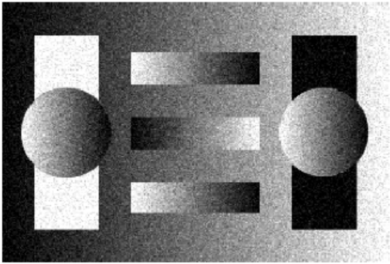
textbooks: Scherzer et al. 2008, Ito and Jin 2015, Benning and Burger 2018

Example: Regularizers

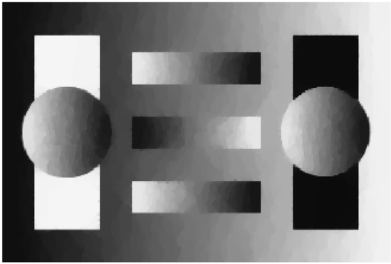
- Tikhonov regularization: $\mathcal{R}(x) = \frac{1}{2}\|x\|_2^2$
- H^1 squared semi-norm: $\mathcal{R}(x) = \frac{1}{2}\|\nabla x\|_2^2$

Example: Regularizers

- Tikhonov regularization: $\mathcal{R}(x) = \frac{1}{2}\|x\|_2^2$
- H^1 squared semi-norm: $\mathcal{R}(x) = \frac{1}{2}\|\nabla x\|_2^2$
- Total Variation $\mathcal{R}(x) = \|\nabla x\|_1$ Rudin, Osher, Fatemi 1992



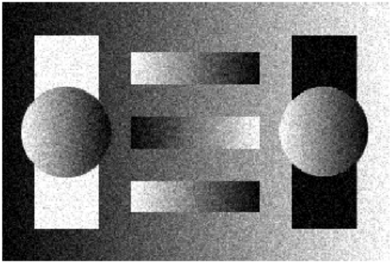
Noisy image



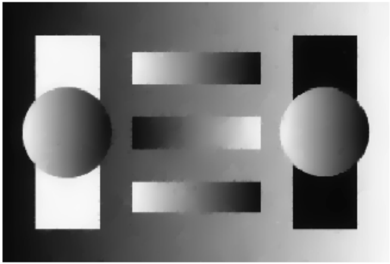
TV denoised image

Example: Regularizers

- Tikhonov regularization: $\mathcal{R}(x) = \frac{1}{2}\|x\|_2^2$
- H^1 squared semi-norm: $\mathcal{R}(x) = \frac{1}{2}\|\nabla x\|_2^2$
- Total Variation $\mathcal{R}(x) = \|\nabla x\|_1$ Rudin, Osher, Fatemi 1992
- Total Generalized Variation
  $\mathcal{R}(x) = \inf_v \|\nabla x - v\|_1 + \beta\|\nabla v\|_1$ Bredies, Kunisch, Pock 2010



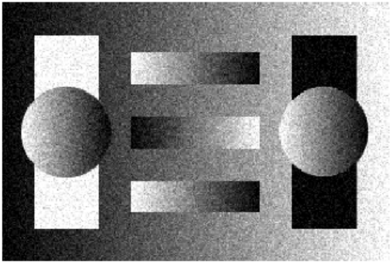
Noisy image



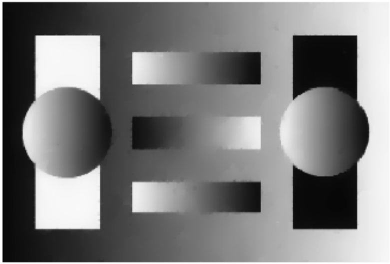
TGV^2 denoised image

Example: Regularizers

- Tikhonov regularization: $\mathcal{R}(x) = \frac{1}{2}\|x\|_2^2$
- H^1 squared semi-norm: $\mathcal{R}(x) = \frac{1}{2}\|\nabla x\|_2^2$
- Total Variation $\mathcal{R}(x) = \|\nabla x\|_1$ Rudin, Osher, Fatemi 1992
- Total Generalized Variation
  $\mathcal{R}(x) = \inf_v \|\nabla x - v\|_1 + \beta\|\nabla v\|_1$ Bredies, Kunisch, Pock 2010



Noisy image

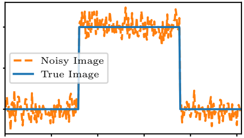


TGV^2 denoised image

How to choose the regularization?

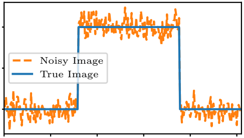
More “complicated” regularizers

$$\min_x \; \frac{1}{2}\|Ax - y\|_2^2 + \alpha\left(\underbrace{\sum_j \|(\nabla x)_j\|_2}_{=\mathrm{TV}(x)} \qquad\qquad\right)$$



More "complicated" regularizers

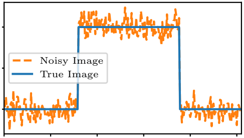
$$\min_x \frac{1}{2}\|Ax-y\|_2^2+\alpha\left(\underbrace{\sum_j \sqrt{\|(\nabla x)_j\|_2^2+\nu^2}}_{\approx \mathrm{TV}(x)}+\frac{\xi}{2}\|x\|_2^2\right)$$



- Smooth and strongly convex
- Solution depends on choices of α, ν and ξ

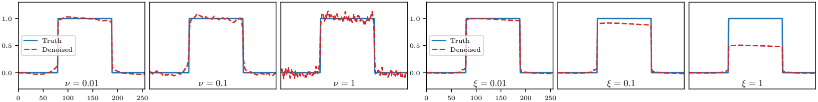
More “complicated” regularizers

$$\min_x \frac{1}{2}\|Ax-y\|_2^2+\alpha\left(\underbrace{\sum_j \sqrt{\|(\nabla x)_j\|_2^2+\nu^2}}_{\approx \mathrm{TV}(x)}+\frac{\xi}{2}\|x\|_2^2\right)$$



- Smooth and strongly convex
- Solution depends on choices of α, ν and ξ

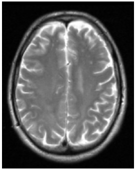
Vary ν ($\alpha = 1$, $\xi = 10^{-3}$) Vary ξ ($\alpha = 1$, $\nu = 10^{-3}$)



How to choose all these parameters?

Example: Magnetic Resonance Imaging (MRI)

MRI scanner

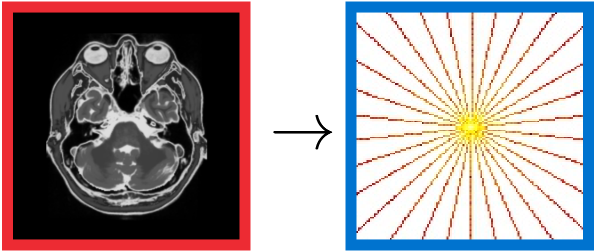


T_2^*

Continuous model: Fourier transform

$$Ax(s) = \int_{\mathbb{R}^2} x(s) \exp(-ist) dt$$

Dicrete model: $A = SF \in \mathbb{C}^{n \times N}$



Solution **not unique**.

Example: MRI reconstruction

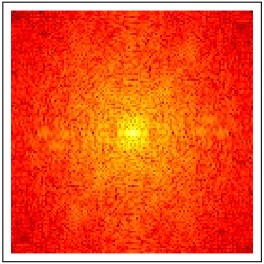
Compressed Sensing MRI:

$A = S \circ F$ Lustig, Donoho, Pauly 2007

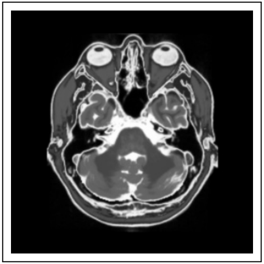
Fourier transform F, sampling $Sw = (w_i)_{i \in \Omega}$

$$\hat{x} \in \arg\min_x \left\{ \sum_{i \in \Omega} |(Fx)_i - y_i|^2 + \lambda \|\nabla x\|_1 \right\}$$

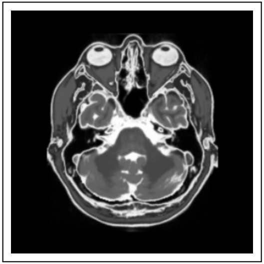
Miki Lustig



sampling $S^* y$



$\lambda = 0$



$\lambda = 1$

Example: MRI reconstruction

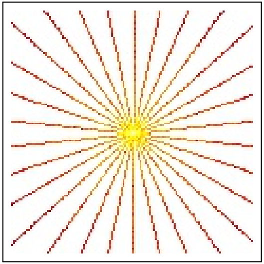
Compressed Sensing MRI:

$A = S \circ F$ Lustig, Donoho, Pauly 2007

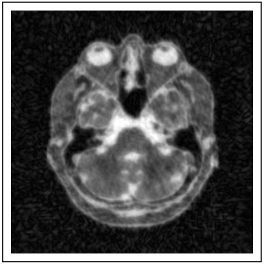
Fourier transform F, sampling $Sw = (w_i)_{i \in \Omega}$

$$\hat{x} \in \arg\min_{x} \left\{ \sum_{i \in \Omega} |(Fx)_i - y_i|^2 + \lambda \|\nabla x\|_1 \right\}$$

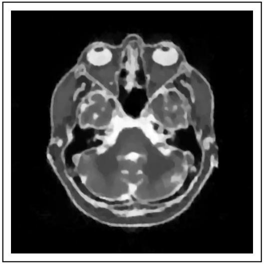
Miki Lustig



sampling $S^* y$



$\lambda = 0$



$\lambda = 10^{-4}$

Example: MRI reconstruction

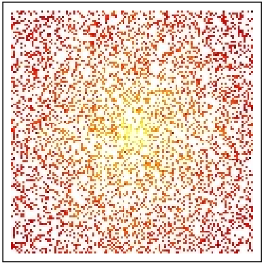
Compressed Sensing MRI:

$A = S \circ F$ Lustig, Donoho, Pauly 2007

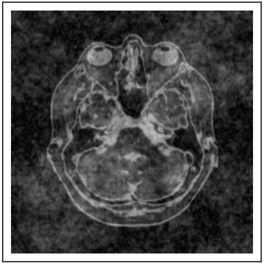
Fourier transform F, sampling $Sw = (w_i)_{i \in \Omega}$

$$\hat{x} \in \arg\min_{x} \left\{ \sum_{i \in \Omega} |(Fx)_i - y_i|^2 + \lambda \|\nabla x\|_1 \right\}$$

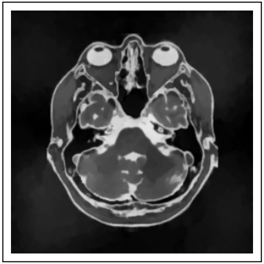
Miki Lustig



sampling $S^* y$



$\lambda = 0$



$\lambda = 10^{-4}$

Example: MRI reconstruction

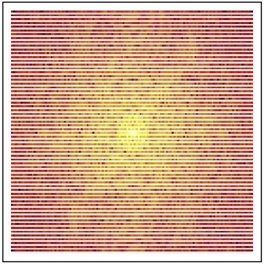
Compressed Sensing MRI:

$A = S \circ F$ Lustig, Donoho, Pauly 2007

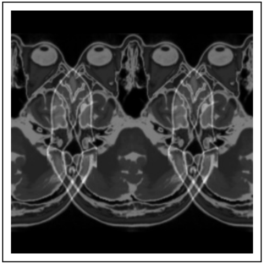
Fourier transform F, sampling $Sw = (w_i)_{i \in \Omega}$

$$\hat{x} \in \arg\min_x \left\{ \sum_{i \in \Omega} |(Fx)_i - y_i|^2 + \lambda \|\nabla x\|_1 \right\}$$

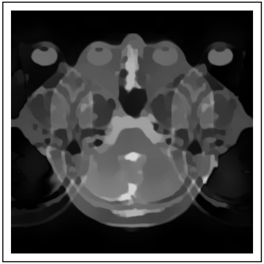
Miki Lustig



sampling $S^* y$



$\lambda = 0$



$\lambda = 10^{-3}$

How to choose the sampling Ω? Is there an optimal sampling?

Does a good sampling depend on $\mathcal{R}$ and λ?

Motivation

- **Inverse problems** can be solved via **variational regularization**

Motivation

- **Inverse problems** can be solved via **variational regularization**
- These models have **a number of parameters**: regularizer, regularization parameter, sampling, smoothness, strong convexity ...

Motivation

- **Inverse problems** can be solved via **variational regularization**
- These models have **a number of parameters**: regularizer, regularization parameter, sampling, smoothness, strong convexity ...
- Some of these parameters have underlying theory and heuristics but are generally still **difficult to choose** in practice

Bilevel Learning

Bilevel learning for inverse problems

$$\hat{x} \in \arg\min_x \{\mathcal{D}(Ax, y) + \lambda \mathcal{R}(x)\}$$

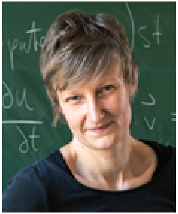
Bilevel learning for inverse problems

Upper level (learning):
Given $(x^\dagger, y), y = Ax^\dagger + \varepsilon$, solve

$$\min_{\lambda \geq 0, \hat{x}} \|\hat{x} - x^\dagger\|_2^2$$

Lower level (solve inverse problem):

$$\hat{x} \in \arg\min_x \{\mathcal{D}(Ax, y) + \lambda \mathcal{R}(x)\}$$



Carola Schönlieb

von Stackelberg 1934, Kunisch and Pock 2013, De los Reyes and Schönlieb 2013

Bilevel learning for inverse problems

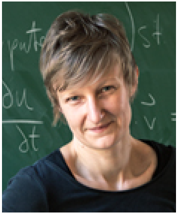
Upper level (learning):

Given $(x_i^\dagger, y_i)_{i=1}^n, y_i = Ax_i^\dagger + \varepsilon_i$, solve

$$\min_{\lambda \geq 0, \hat{x}_i} \frac{1}{n} \sum_{i=1}^n \|\hat{x}_i - x_i^\dagger\|_2^2$$

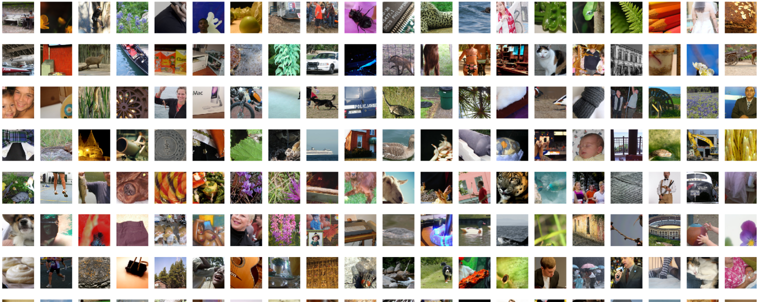
Lower level (solve inverse problem):

$$\hat{x}_i \in \arg\min_x \{\mathcal{D}(Ax, y_i) + \lambda \mathcal{R}(x)\}$$



Carola Schönlieb

von Stackelberg 1934, Kunisch and Pock 2013, De los Reyes and Schönlieb 2013



Inexact Algorithms for Bilevel Learning

Bilevel learning: Reduced formulation

Upper level:

$$\min_{\lambda \geq 0, \hat{x}} \|\hat{x} - x^\dagger\|_2^2$$

Lower level:

$$\hat{x} = \arg\min_x \{\mathcal{D}(Ax, y) + \lambda \mathcal{R}(x)\}$$

Bilevel learning: Reduced formulation

Upper level:

$$\min_{\lambda \geq 0, \hat{x}} U(\hat{x})$$

Lower level:

$$\hat{x} = \arg\min_{x} \{\mathcal{D}(Ax, y) + \lambda \mathcal{R}(x)\}$$

Bilevel learning: Reduced formulation

Upper level:

$$\min_{\lambda \geq 0, \hat{x}} U(\hat{x})$$

Lower level:

$$\hat{x} = \arg\min_x L(x, \lambda)$$

Bilevel learning: Reduced formulation

Upper level: $$\min_{\lambda \geq 0, \hat{x}} U(\hat{x})$$

Lower level:
$$x(\lambda) := \hat{x} = \arg\min_x L(x, \lambda)$$

Reduced formulation: $$\min_{\lambda \geq 0} U(x(\lambda)) =: \tilde{U}(\lambda)$$

Bilevel learning: Reduced formulation

Upper level: $$\min_{\lambda \geq 0, \hat{x}} U(\hat{x})$$

Lower level:

$$x(\lambda) := \hat{x} = \arg\min_x L(x, \lambda) \quad \Leftrightarrow \quad \partial_x L(x(\lambda), \lambda) = 0$$

Reduced formulation: $$\min_{\lambda \geq 0} U(x(\lambda)) =: \tilde{U}(\lambda)$$

Bilevel learning: Reduced formulation

Upper level: $$\min_{\lambda \geq 0, \hat{x}} U(\hat{x})$$

Lower level:
$$x(\lambda) := \hat{x} = \arg\min_x L(x, \lambda) \quad \Leftrightarrow \quad \partial_x L(x(\lambda), \lambda) = 0$$

Reduced formulation: $$\min_{\lambda \geq 0} U(x(\lambda)) =: \tilde{U}(\lambda)$$

$$0 = \partial_x^2 L(x(\lambda), \lambda) x'(\lambda) + \partial_\lambda \partial_x L(x(\lambda), \lambda) \quad \Leftrightarrow \quad x'(\lambda) = -B^{-1} A$$

Bilevel learning: Reduced formulation

Upper level: $$\min_{\lambda \geq 0, \hat{x}} U(\hat{x})$$

Lower level:
$$x(\lambda) := \hat{x} = \arg\min_{x} L(x, \lambda) \quad \Leftrightarrow \quad \partial_x L(x(\lambda), \lambda) = 0$$

Reduced formulation: $$\min_{\lambda \geq 0} U(x(\lambda)) =: \tilde{U}(\lambda)$$

$$0 = \partial_x^2 L(x(\lambda), \lambda) x'(\lambda) + \partial_\lambda \partial_x L(x(\lambda), \lambda) \quad \Leftrightarrow \quad x'(\lambda) = -B^{-1} A$$

$$\nabla \tilde{U}(\lambda) = (x'(\lambda))^* \nabla U(x(\lambda))$$

Bilevel learning: Reduced formulation

Upper level: $$\min_{\lambda\geq 0,\hat{x}} U(\hat{x})$$

Lower level:
$$x(\lambda) := \hat{x} = \arg\min_x L(x,\lambda) \quad \Leftrightarrow \quad \partial_x L(x(\lambda),\lambda) = 0$$

Reduced formulation: $\min_{\lambda\geq 0} U(x(\lambda)) =: \tilde{U}(\lambda)$

$$0 = \partial_x^2 L(x(\lambda),\lambda)x'(\lambda) + \partial_\lambda \partial_x L(x(\lambda),\lambda) \quad \Leftrightarrow \quad x'(\lambda) = -B^{-1}A$$

$$\begin{aligned}\nabla \tilde{U}(\lambda) &= (x'(\lambda))^* \nabla U(x(\lambda)) \\ &= -A^* B^{-1} \nabla U(x(\lambda)) = -A^* w\end{aligned}$$

where w solves $Bw = \nabla U(x(\lambda))$.

Algorithm for Bilevel learning

Upper level: $\min_{\lambda \geq 0, \hat{x}} U(\hat{x})$

Lower level: $x(\lambda) := \arg\min_x L(x, \lambda)$

Reduced formulation: $\min_{\lambda \geq 0} U(x(\lambda)) =: \tilde{U}(\lambda)$

- Solve reduced formulation via L-BFGS-B Nocedal and Wright 2000
- Compute gradients: Given λ
  (1) Compute $x(\lambda)$, e.g. via PDHG Chambolle and Pock 2011
  (2) Solve $Bw = \nabla U(x(\lambda))$, $B := \partial_x^2 L(x(\lambda), \lambda)$ e.g. via CG
  (3) Compute $\nabla \tilde{U}(\lambda) = -A^* w$, $A := \partial_\lambda \partial_x L(x(\lambda), \lambda)$

Algorithm for Bilevel learning

Upper level: $\min_{\lambda \geq 0, \hat{x}} U(\hat{x})$

Lower level: $x(\lambda) := \arg\min_x L(x, \lambda)$

Reduced formulation: $\min_{\lambda \geq 0} U(x(\lambda)) =: \tilde{U}(\lambda)$

- Solve reduced formulation via L-BFGS-B Nocedal and Wright 2000
- Compute gradients: Given λ
  1. Compute $x(\lambda)$, e.g. via PDHG Chambolle and Pock 2011
  2. Solve $Bw = \nabla U(x(\lambda))$, $B := \partial_x^2 L(x(\lambda), \lambda)$ e.g. via CG
  3. Compute $\nabla \tilde{U}(\lambda) = -A^* w$, $A := \partial_\lambda \partial_x L(x(\lambda), \lambda)$

This approach has a number of problems:

- $x(\lambda)$ has to be computed
- Derivative assumes $x(\lambda)$ is exact minimizer
- Large system of linear equations has to be solved

How to solve Bilevel Learning Problems?

- Most people: Ignore “problems”, just compute it. e.g. Sherry et al. 2020
- Semi-smooth Newton: similar fundamental problems Kunisch and Pock 2013
- Replace lower level problem by finite number of iterations of algorithms: not bilevel anymore Ochs et al. 2015

How to solve Bilevel Learning Problems?

- Most people: Ignore “problems”, just compute it. e.g. Sherry et al. 2020
- Semi-smooth Newton: similar fundamental problems Kunisch and Pock 2013
- Replace lower level problem by finite number of iterations of algorithms: not bilevel anymore Ochs et al. 2015

Use algorithm that acknowledges difficulties:
e.g. inexact DFO Ehrhardt and Roberts 2021

Dynamic Accuracy Derivative Free Optimization

$$\min_{\theta} f(\theta)$$

Key idea: Use f_ϵ:

$$|f(\theta) - f_\epsilon(\theta)| < \epsilon$$

Accuracy as low as possible, but as high as necessary.

E.g. if

$$f_{\epsilon^{k+1}}(\theta^{k+1}) < f_{\epsilon^k}(\theta^k) - \epsilon^k - \epsilon^{k+1},$$

then

$$f(\theta^{k+1}) < f(\theta^k)$$

Dynamic Accuracy Derivative Free Optimization

$$\min_{\theta} f(\theta)$$

For $k = 0, 1, 2, \ldots$

1) Sample f_{ϵ^k} in a neighbourhood of θ_k
2) Build model $m_k(\theta) \approx f_{\epsilon^k}$
3) Minimise m_k around θ_k to get θ_{k+1}
4) If model decrease is sufficient compared to function error: accept step

Algorithm 1 Dynamic accuracy DFO algorithm for (22).

Inputs: Starting point $\theta^0 \in \mathbb{R}^n$, initial trust-region radius $0 < \Delta^0 \le \Delta_{\max}$.

Parameters: strictly positive values $\Delta_{\max}, \gamma_{\text{dec}}, \gamma_{\text{inc}}, \eta_1, \eta_2, \eta_1', \epsilon$ satisfying $\gamma_{\text{dec}} < 1 < \gamma_{\text{inc}}$, $\eta_1 \le \eta_2 < 1$, and $\eta_1' < \min(\eta_1, 1 - \eta_2)/2$.

1: Select an arbitrary interpolation set and construct m^0 (26).
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: **repeat**
4: Evaluate $\widetilde{f}(\theta^k)$ to sufficient accuracy that (32) holds with η_1' (using s^k from the previous iteration of this inner repeat/until loop). Do nothing in the first iteration of this repeat/until loop.
5: **if** $\|g^k\| \le \epsilon$ **then**
6: By replacing Δ^k with $\gamma_{\text{dec}}^i \Delta^k$ for $i = 0, 1, 2, \ldots$, find m^k and Δ^k such that m^k is fully linear in $B(\theta^k, \Delta^k)$ and $\Delta^k \le \|g^k\|$. *[criticality phase]*
7: **end if**
8: Calculate s^k by (approximately) solving (27).
9: **until** the accuracy in the evaluation of $\widetilde{f}(\theta^k)$ satisfies (32) with η_1' *[accuracy phase]*
10: Evaluate $\widetilde{r}(\theta^k + s^k)$ so that (32) is satisfied with η_1' for $\widetilde{f}(\theta^k + s^k)$, and calculate $\widetilde{\rho}^k$ (29).
11: Set θ^{k+1} and Δ^{k+1} as:

$$\theta^{k+1} = \begin{cases} \theta^k + s^k, & \widetilde{\rho}^k \ge \eta_2, \text{ or } \widetilde{\rho}^k \ge \eta_1 \text{ and } m^k \\ & \text{fully linear in } B(\theta^k, \Delta^k), \\ \theta^k, & \text{otherwise,} \end{cases} \tag{33}$$

and

$$\Delta^{k+1} = \begin{cases} \min(\gamma_{\text{inc}} \Delta^k, \Delta_{\max}), & \widetilde{\rho}^k \ge \eta_2, \\ \Delta^k, & \widetilde{\rho}^k < \eta_2 \text{ and } m^k \text{ not} \\ & \text{fully linear in} B(\theta^k, \Delta^k), \\ \gamma_{\text{dec}} \Delta^k, & \text{otherwise.} \end{cases} \tag{34}$$

12: If $\theta^{k+1} = \theta^k + s^k$, then build m^{k+1} by adding θ^{k+1} to the interpolation set (removing an existing point). Otherwise, set $m^{k+1} = m^k$ if m^k is fully linear in $B(\theta^k, \Delta^k)$, or form m^{k+1} by making m^k fully linear in $B(\theta^{k+1}, \Delta^{k+1})$.
13: **end for**

Theorem Ehrhardt and Roberts 2021

If f is sufficiently smooth and bounded below, then the algorithm is globally convergent in the sense that

$$\lim_{k \to \infty} \|\nabla f(\theta_k)\| = 0 \,.$$

1D Denoising Problem (learn α, ν and ξ) Ehrhardt and Roberts 2021

$$\min_{\theta} \left\{ \frac{1}{2} \sum_i \|x_i(\theta) - x_i\|_2^2 + \beta \left(\frac{L(\theta)}{\kappa(\theta)} \right)^2 \right\}$$

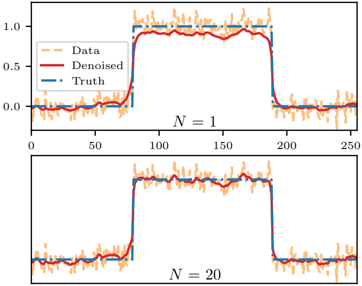
$$x_i(\theta) = \arg\min_x \frac{1}{2}\|x - y_i\|_2^2 + \alpha \left(\sum_j \sqrt{\|(\nabla x)_j\|_2^2 + \nu^2} + \frac{\xi}{2}\|x\|_2^2 \right)$$

1D Denoising Problem (learn α, ν and ξ) Ehrhardt and Roberts 2021

$$\min_{\theta} \left\{ \frac{1}{2} \sum_i \|x_i(\theta) - x_i\|_2^2 + \beta \left(\frac{L(\theta)}{\kappa(\theta)} \right)^2 \right\}$$

$$x_i(\theta) = \arg\min_x \frac{1}{2}\|x - y_i\|_2^2 + \alpha \left(\sum_j \sqrt{\|(\nabla x)_j\|_2^2 + \nu^2} + \frac{\xi}{2}\|x\|_2^2 \right)$$

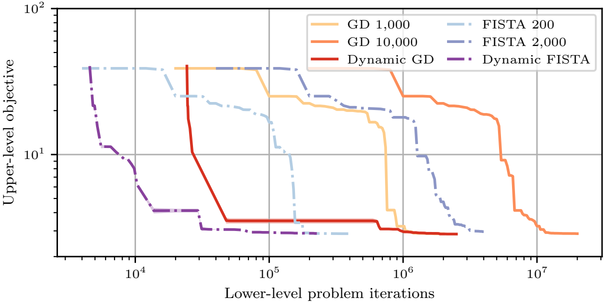
With more evaluations of $f(\theta)$, the parameter choices give better reconstructions:



Reconstruction of x_1 after N evaluations of $f(\theta)$

1D Denoising Problem (learn α, ν and ξ) Ehrhardt and Roberts 2021

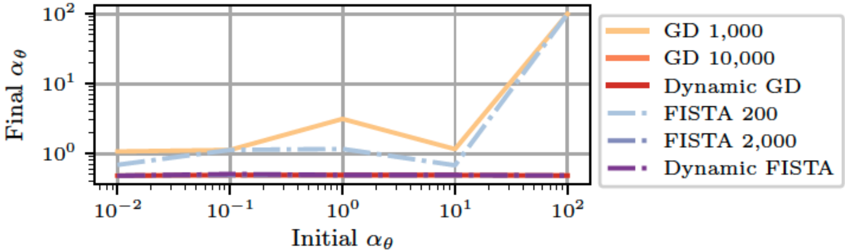
Dynamic accuracy is faster than “fixed accuracy” (at least 10x speedup):



Objective value $f(\theta)$ vs. computational effort

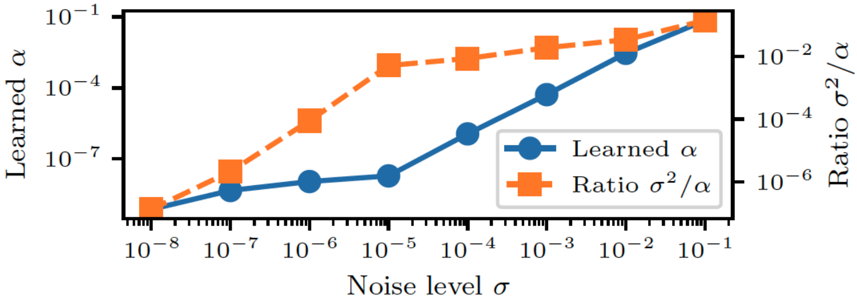
1D Denoising Problem Ehrhardt and Roberts 2021

Always learns the same parameter for sufficient accuracy.



Robustness to initialization

Denoising Problem (learn α, ν and ξ) Ehrhardt and Roberts 2021



Bilevel learning is a convergent regularization?

Learn sampling pattern in MRI

Some important works on sampling for MRI

Uninformed

- Cartesian, radial, variable density ... e.g. Lustig et al. '07
  - ✓ simple to implement
  - ✗ not tailored to application or reconstruction method
- compressed sensing e.g. Candes and Romberg '07, Kutyniok and Lim '18
  - ✓ mathematical guarantees
  - ✗ limited to sparse signals and sparsity promoting regularizers

Some important works on sampling for MRI

Uninformed

- Cartesian, radial, variable density ... e.g. Lustig et al. '07
  - ✓ simple to implement
  - ✗ not tailored to application or reconstruction method
- compressed sensing e.g. Candes and Romberg '07, Kutyniok and Lim '18
  - ✓ mathematical guarantees
  - ✗ limited to sparse signals and sparsity promoting regularizers

Learned

- **Largest Fourier coefficients** of training set Knoll et al. '11
  - ✓ simple to implement, computationally efficient
  - ✗ not tailored to reconstruction method
- **greedy**: iteratively select "best" sample e.g. Gözcü et al. '18
  - ✓ adaptive to dataset, reconstruction method
  - ✗ only discrete values; computationally heavy
- **Deep learning**: e.g. parameters in network Wang et al. '21
  - ✓ realistic and easy to implement sampling patterns; end-to-end
  - ✗ limited to neural network reconstruction

Learn sampling pattern in MRI

Lower level (MRI reconstruction):

$$x_i(\lambda, s) = \arg\min_x \left\{ \sum_{j=1}^{N} s_j^2 |(Fx - y_i)_j|^2 + \lambda \mathcal{R}(x) \right\} \quad s_i \in \{0, 1\}$$

Sherry et al. 2020

Learn sampling pattern in MRI

Upper level (learning):
Given **training data** $(x_i^\dagger, y_i)_{i=1}^n$, solve

$$\min_{\lambda \geq 0, s \in \{0,1\}^m} \frac{1}{n} \sum_{i=1}^{n} \|x_i(\lambda, s) - x_i^\dagger\|_2^2$$

Lower level (MRI reconstruction):

$$x_i(\lambda, s) = \arg\min_x \left\{ \sum_{j=1}^{N} s_j^2 |(Fx - y_i)_j|^2 + \lambda \mathcal{R}(x) \right\} \quad s_i \in \{0, 1\}$$

Sherry et al. 2020

Learn sampling pattern in MRI

Upper level (learning):
Given **training data** $(x_i^\dagger, y_i)_{i=1}^n$, solve

$$\min_{\lambda \geq 0, s \in \{0,1\}^m} \frac{1}{n} \sum_{i=1}^n \|x_i(\lambda, s) - x_i^\dagger\|_2^2 + \beta_1 \sum_{j=1}^m s_j$$

Lower level (MRI reconstruction):

$$x_i(\lambda, s) = \arg\min_x \left\{ \sum_{j=1}^N s_j^2 |(Fx - y_i)_j|^2 + \lambda \mathcal{R}(x) \right\} \quad s_i \in \{0, 1\}$$

Sherry et al. 2020

Learn sampling pattern in MRI

Upper level (learning):
Given **training data** $(x_i^\dagger, y_i)_{i=1}^n$, solve

$$\min_{\lambda\geq 0, s\in[0,1]^m} \frac{1}{n}\sum_{i=1}^n \|x_i(\lambda, s) - x_i^\dagger\|_2^2 + \beta_1 \sum_{j=1}^m s_j$$

Lower level (MRI reconstruction):

$$x_i(\lambda, s) = \arg\min_x \left\{ \sum_{j=1}^N s_j^2 |(Fx - y_i)_j|^2 + \lambda \mathcal{R}(x) \right\} \quad s_i \in [0,1]$$

Sherry et al. 2020

Learn sampling pattern in MRI

Upper level (learning):

Given **training data** $(x_i^\dagger, y_i)_{i=1}^n$, solve

$$\min_{\lambda \geq 0, s \in [0,1]^m} \frac{1}{n} \sum_{i=1}^{n} \| x_i(\lambda, s) - x_i^\dagger \|_2^2 + \beta_1 \sum_{j=1}^{m} s_j + \beta_2 \sum_{j=1}^{m} s_j (1 - s_j)$$

Lower level (MRI reconstruction):

$$x_i(\lambda, s) = \arg\min_x \left\{ \sum_{j=1}^{N} s_j^2 |(Fx - y_i)_j|^2 + \lambda \mathcal{R}(x) \right\} \quad s_i \in [0, 1]$$

Sherry et al. 2020

Warm up

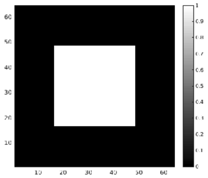
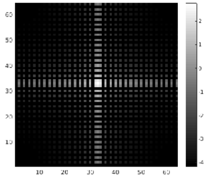
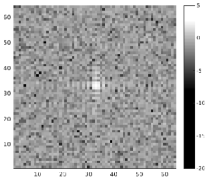


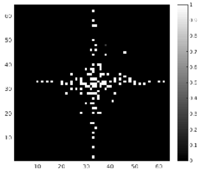
Figure: Discrete 2d bump



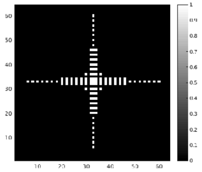
(a) Original data: $\log|y|$



(b) Noisy data: $\log|\tilde{y}|$



(c) Learned sampling pattern



(d) Largest 2.76% Fourier Coefficients

Warm up

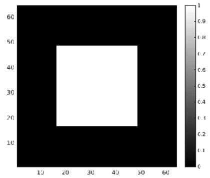
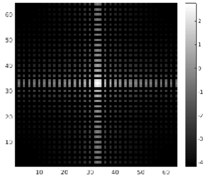
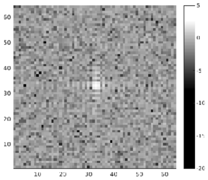


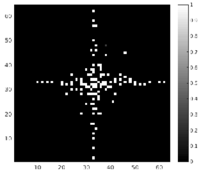
Figure: Discrete 2d bump



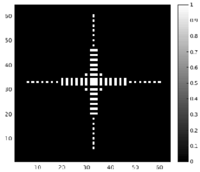
(a) Original data: $\log |y|$



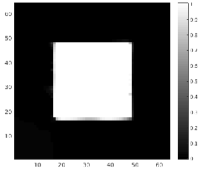
(b) Noisy data: $\log |\tilde{y}|$



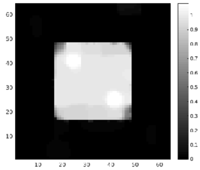
(c) Learned sampling pattern



(d) Largest 2.76% Fourier Coefficients



(e) Learned sampling pattern



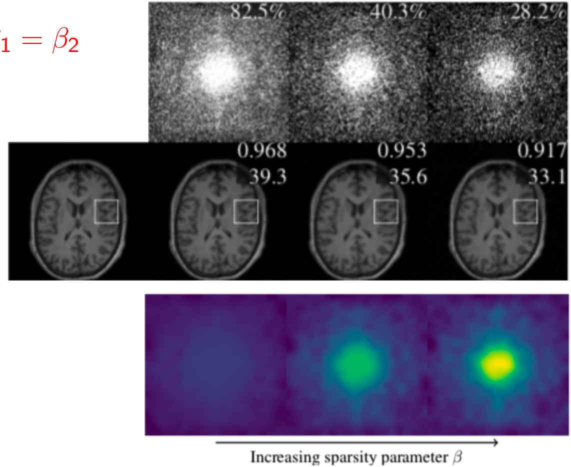
(f) Largest 2.76% Fourier Coefficients

Increasing sparsity Sherry et al. 2020

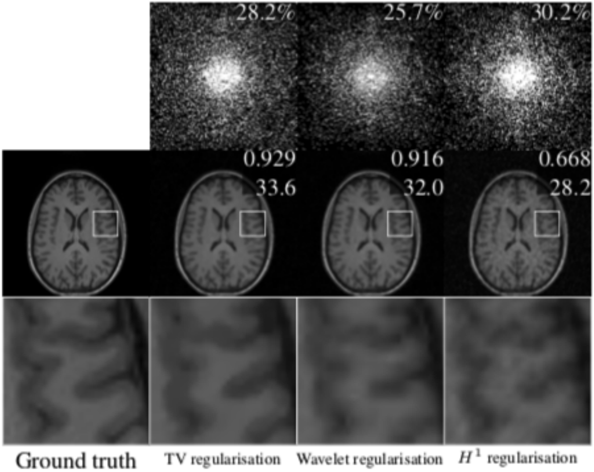
Reminder: **Upper level** (learning)

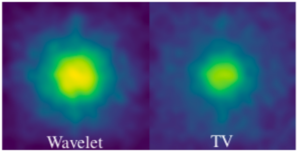
$$\min_{\lambda \geq 0, s \in [0,1]^m} \frac{1}{n} \sum_{i=1}^{n} \|x_i(\lambda, s) - x_i^\dagger\|_2^2 + \beta_1 \sum_{j=1}^{m} s_j + \beta_2 \sum_{j=1}^{m} s_j(1 - s_j)$$

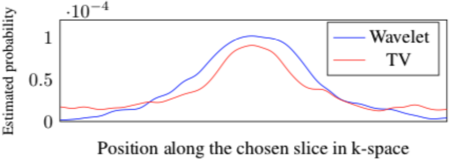
$\beta = \beta_1 = \beta_2$



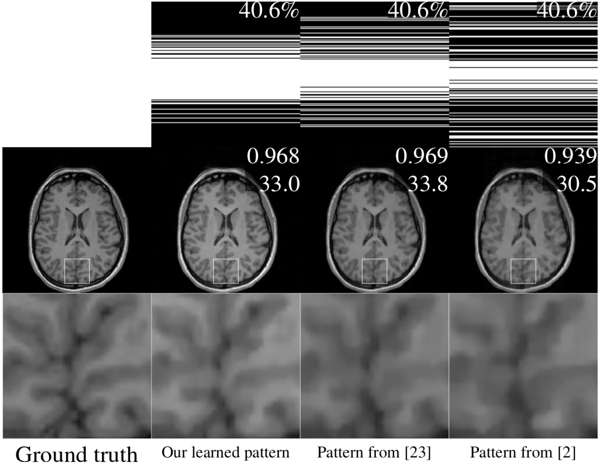
Compare regularizers Sherry et al. 2020







Compare Cartesian samplings Sherry et al. 2020



	Line sampling (40.6%)	Free pattern (34.7%)
Our method	4192	6494
The method from [23]	12087	$3.90 \cdot 10^8$

number of lower-level solves

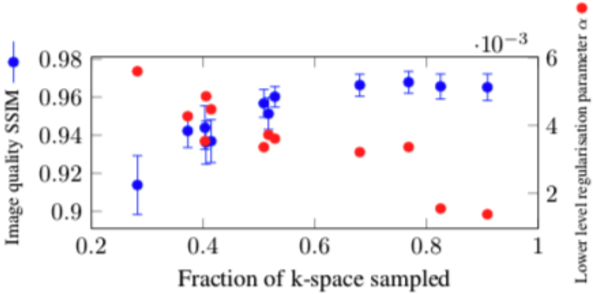
"ours" = Sherry et al. 2020

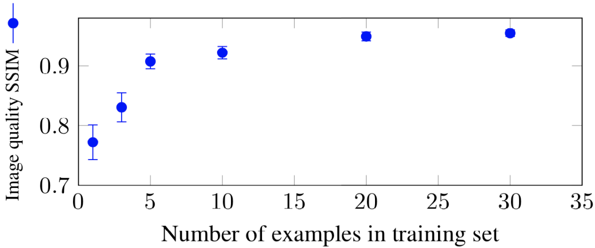
[23] = Gözcü et al. 2018

[2] = Lustig et al. 2007

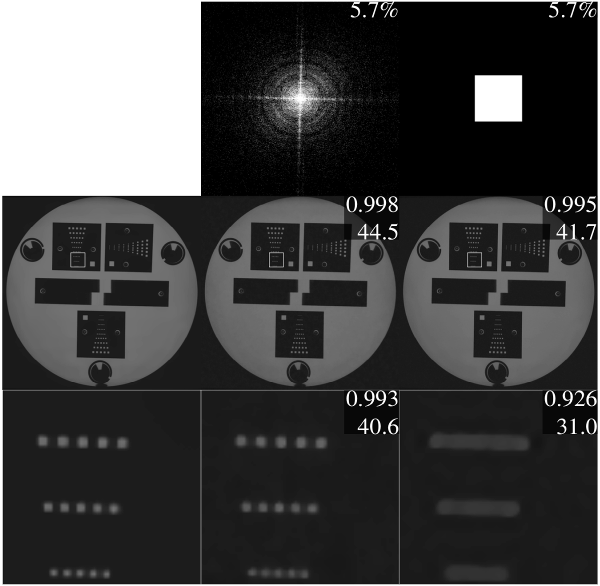
regularizer = TV

More insights: sampling and number of data Sherry et al. 2020





High resolution imaging: 1024^2 Sherry et al. 2020



Conclusions

- **Bilevel learning**: supervised learning framework to learn parameters in variational regularization
- **Optimization** plays a key role in bilevel learning
  - Dynamic accuracy: no need to specify number of iterations
  - Make learning surprisingly robust
- **Learned sampling** better than generic sampling
  - "Optimal" sampling depends on regularizer
  - Very little data needed

Conclusions

- **Bilevel learning**: supervised learning framework to learn parameters in variational regularization
- **Optimization** plays a key role in bilevel learning
  - Dynamic accuracy: no need to specify number of iterations
  - Make learning surprisingly robust
- **Learned sampling** better than generic sampling
  - "Optimal" sampling depends on regularizer
  - Very little data needed

Future work

- **Stochastic** algorithms (like stochastic gradient descent etc)
- **Nonsmooth or nonconvex** lower-level problems
- **Inexact gradient** methods
- **Neural network** regularization