Solving ill-posed inverse problems using iterative deep neural networks

Jonas Adler and Ozan Öktem

Inverse Problems **33***(2017), 124007*

Speaker: Haibo Li

2021.9.1

Contents

Inverse Problems

- Reconstructing a signal $f_{\text{true}} \in X$ from data $g \in Y$ where

$$g = \mathcal{T}(f_{\text{true}}) + \delta g. \tag{1.1}$$

Forward operator $\mathcal{T} \colon X \to Y$, noise vector $\delta g \in Y$

- Minimize the miss-fit against data:

$$f \to \mathcal{L}\big(\mathcal{T}(f), g\big) \tag{1.2}$$

where $\mathcal{L} \colon Y \times Y \to \mathbb{R}$ is a suitable affine transformation of the data log-likelihood.
Typical choices of $\mathcal{T}$ is ill posed: solution unstable with respect to g in the sense that small changes to data results in large changes to a reconstruction.

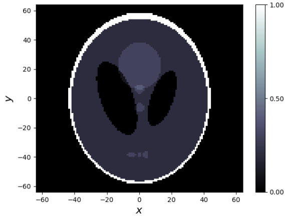
Classical regularization

- Approximate $\mathcal{T}^{\dagger}$: construct a pseudo-inverse to $\mathcal{T}$ in a certain sense.
- Iterative regularization: an iteration scheme for minimizing (1.2), stop the iterates early.
- Variational regularization:

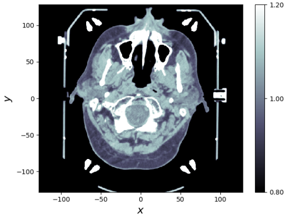
$$\min_{f \in X} \left[\mathcal{L}\big(\mathcal{T}(f), g\big) + \lambda\, \mathcal{S}(f) \right] \quad \text{for a fixed } \lambda \geq 0. \tag{1.3}$$

Fidelity term $\mathcal{L}\big(\mathcal{T}(f), g\big)$ (encoding forward operator), regularization term $\mathcal{S}(f)$ (encoding a priori knowledge of f_{true}), regularization parameter λ.

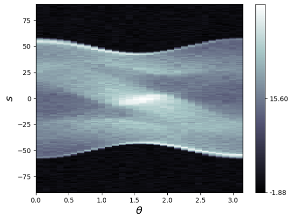
Example: Computed tomography (CT)



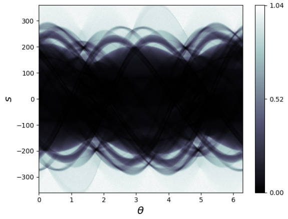
(a)



(b)



(c)



(d)

Figure: (a) Shepp-Logan phantom; (b) slice of head phantom; (c)(d) simulated tomographic

Machine learning approaches

- Reconstructing a mapping $\mathcal{T}_{\mathbf{\Theta}}^{\dagger} : Y \to X$ satisfying:

  $$\mathcal{T}_{\mathbf{\Theta}}^{\dagger}(g) \approx f_{\mathsf{true}}.$$

  The "learning" part refers to choosing "optimal" parameters $\mathbf{\Theta} \in Z$.

- Supervised learning: training data (g, f) are $(Y \times X)$–valued i.i.d random variable with a known probability density μ.
  Loss function:

  $$\mathrm{L}(\mathbf{\Theta}) := \mathbb{E}_{\mu}\Big[\big\|\mathcal{T}_{\mathbf{\Theta}}^{\dagger}(\mathrm{g}) - \mathsf{f}\big\|_{X}^{2}\Big].$$

- Unsupervised learning: training data g are Y–valued i.i.d random variable with a known probability density μ.
  Loss function:

  $$\mathrm{L}(\mathbf{\Theta}) := \mathbb{E}_{\mu}\Big[\mathcal{L}\Big(\mathcal{T}\big(\mathcal{T}_{\mathbf{\Theta}}^{\dagger}(\mathrm{g})\big), \mathrm{g}\Big) + \mathcal{S}\big(\mathcal{T}_{\mathbf{\Theta}}^{\dagger}(\mathrm{g})\big)\Big].$$

Machine learning approaches

- Fully learned reconstruction: parameter space Z very high dimensional, learning $\mathcal{T}^{\dagger}_{\mathbf{\Theta}}$ from data without using any knowledge of physics quickly becomes in-feasible.
- Sequential data and knowledge driven reconstruction: separate the learned components from a part that encodes knowledge about $\mathcal{T}$ and the data manifold,
$$\mathcal{T}^{\dagger}_{\mathbf{\Theta}} = \mathcal{B}_{\mathbf{\Theta}} \circ \mathcal{A} \circ \mathcal{C}_{\mathbf{\Theta}} \tag{1.4}$$
where $\mathcal{A}\colon Y \to X$ is a known component that encodes knowledge about $\mathcal{T}$, $\mathcal{B}_{\mathbf{\Theta}}\colon X \to X$, $\mathcal{C}_{\mathbf{\Theta}}\colon Y \to Y$ are learned components.
- Learning for variational reconstruction; Learning an iterative reconstruction scheme.

Motivation

The regularized solution approximates the true signal to be recovered:

$$\arg\min_{f\in X} \mathrm{E}(f) \approx \arg\min_{f\in X} \Big[\mathcal{L}\big(\mathcal{T}(f), g\big) + \lambda\, \mathcal{S}(f)\Big]. \tag{2.1}$$

where $\mathrm{E}(f) := \mathrm{d}(f, f_{\text{true}})$ and $\mathrm{d}\colon X \times X \to \mathbb{R}$ is the distance functional (typically $\mathrm{E}(f) = \|f - f_{\text{true}}\|_X^2$).

Gradient descent:

right-hand term: $f_i := f_{i-1} - \sigma\Big(\nabla\big[\,\mathcal{L}\big(\mathcal{T}(\cdot), g\big)\big](f_{i-1}) + \lambda[\nabla\mathcal{S}](f_{i-1})\Big)$ (2.2)

left-hand term: $f_{j+1} := f_j - \sigma[\nabla \mathrm{E}](f_j).$ (2.3)

Introduce *learned updating operator* $\Lambda_{\mathbf{\Theta}}\colon X \times X \times X \to X$, satisfying

$$\Lambda_{\mathbf{\Theta}}\Big(f, \nabla\big[\,\mathcal{L}\big(\mathcal{T}(\cdot), g\big)\big](f), \lambda\,\nabla\mathcal{S}(f)\Big) \approx \nabla\mathrm{E}(f).$$

Initial partially learned gradient descent

Algorithm1: Initial partially learned gradient descent

```
1. Select an initial guess $f_0$
2. **for** $i = 1, \dots, I$
3.     $\Delta f_i \leftarrow -\sigma \Lambda_{\boldsymbol{\Theta}} \Big( f_{i-1}, \nabla \big[ \mathcal{L}\big(\mathcal{T}(\cdot), g\big) \big](f_{i-i}), \lambda \nabla \mathcal{S}(f_{i-1}) \Big)$
4.     $f_i \leftarrow f_{i-1} + \Delta f_i$
5. **end for**
6. $\mathcal{T}^{\dagger}_{\boldsymbol{\Theta}}(g) \leftarrow f_I$
```

Modifications to Algorithm1

- The regularization parameter λ and step length σ can be learned from training data.
- Introduce persistent memory $s \in X^M$ that allows Algorithm1 to use information from earlier iterates:

$$\Lambda_{\boldsymbol{\Theta}} \colon X^M \times X \times X \times X \to X^M \times X. \tag{2.4}$$

- Select initial iterate f_0 using suitable pseudo-inverse $\mathcal{T}^\dagger \colon Y \to X$, i.e., $f_0 = \mathcal{T}^\dagger(g)$.

Partially learned gradient descent

Algorithm2: Partially learned gradient descent

```
1. $f_0 \leftarrow \mathcal{T}^\dagger(g)$.
2. Initialize "memory" $s_0 \in X^M$.
3. for $i = 1, \ldots, I$
4.     $(s_i, \Delta f_i) \leftarrow \Lambda_{\mathbf{\Theta}}\Big(s_{i-1}, f_{i-1}, \nabla\big[\mathcal{L}\big(\mathcal{T}(\cdot), g\big)\big](f_{i-1}), \nabla\mathcal{S}(f_{i-1})\Big)$
5.     $f_i \leftarrow f_{i-1} + \Delta f_i$
6. end for
7. $\mathcal{T}^\dagger_{\mathbf{\Theta}}(g) \leftarrow f_I$
```

Learned updating operator

Define a family of affine operators

$$\mathcal{W}_{w_n,b_n} \colon X^{c_{n-1}} \to X^{c_n} \quad \text{for } n = 0, \dots, N, \tag{2.5}$$

parametrized by linear mappings $w_n \colon X^{c_{n-1}} \to X^{c_n}$ (weights) and $b_n \in X^{c_n}$ (biases). A family of non-linear operators (active function)

$$\mathcal{A}_n \colon X^{c_n} \to X^{c_n} \tag{2.6}$$

Define a parametrized family of learned updating operators as

$$\Lambda_{\boldsymbol{\Theta}} := (\mathcal{A}_N \circ \mathcal{W}_{w_N,b_N}) \circ \dots \circ (\mathcal{A}_1 \circ \mathcal{W}_{w_1,b_1})$$

with $\boldsymbol{\Theta} := \big((w_N, b_N), \dots, (w_1, b_1)\big)$. Let $c_0 = M + 3$ and $c_N = M + 1$.

Affine and non-linear operator families

Affine operator in (2.5) is

$$\mathcal{W}_{w_n,b_n} = (\mathcal{W}^1_{w_n,b_n}, \dots, \mathcal{W}^{c_n}_{w_n,b_n})$$

where the components

$$\mathcal{W}^l_{w_n,b_n} \colon X^{c_{n-1}} \to X \quad \text{for } l = 1, \dots, c_n$$

represent the affine transformation for the l:th channel in the n:th layer, and

$$\mathcal{W}^l_{w_n,b_n}(f_1, \dots, f_{c_{n-1}}) = b^l_n + \sum_{j=1}^{c_{n-1}} w^{j,l}_n(f_j)$$

where $w^{j,l}_n : X \to X$ is a channel-wise linear operator

Convolutional neural network (CNN)

Assume $\mathcal{W}^l_{w_n,b_n}$ is translation invariant:

$$\mathcal{W}^l_{w_n,b_n}(f_1,\ldots,f_{c_{n-1}}) = b^l_n + \sum_{j=1}^{c_{n-1}} w^{j,l}_n * f_j$$

where $b^l_n \in \mathbb{R}$ represents the bias and w_n is given as a "matrix" of convolution kernels $w^{j,l}_n \in X$.
(Remark: In the case of tomographic reconstruction, the situation is however different since there are many non-local dependencies (all pixels/voxels on a line contribute to the value of the ray-transform). Thus, applying convolution network architectures to directly learn the reconstruction would be problematic. On the other hand, the scheme outlined in Algorithm 2 includes the forward operator that accounts for these global dependencies.)

Active function: $\text{relu}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{else.} \end{cases}$

Hyper-parameters settings

- Hyper-parameters: the number of layers $N \in \mathbb{N}$, the number of channels $c_1, \dots, c_{N-1} \in \mathbb{N}$ in each layer, the number of iterations I, and the size of the memory M.
- Set $I = 10$ and $M = 5$, use $N = 3$ layers.
  Let w_i be represented by 3×3 pixels convolutions.
  The number of convolutions in each layer was selected as $m_1 = 32$ and $m_2 = 32$.
- Loss function: $L(\mathbf{\Theta}) = \| \mathcal{T}^{\dagger}_{\mathbf{\Theta}}(g) - f_{\text{true}} \|_X^2$.

Partially learned gradient descent

Algorithm3: Partially learned gradient descent

```
1. $s_0 \leftarrow 0$
2. $f_0 \leftarrow \mathcal{T}^\dagger(g)$
3. **for** $i = 1, \dots, I$
4.     $u_i^1 \leftarrow \Big(f_{i-1}, s_{i-1}, \nabla\big[\mathcal{L}\big(\mathcal{T}(\cdot), g\big)\big](f_{i-1}), \nabla\mathcal{S}(f_{i-1})\Big)$
5.     $u_i^2 \leftarrow \mathrm{relu}\big(\mathcal{W}_{w_1,b_1}(u_i^1)\big)$
6.     $u_i^3 \leftarrow \mathrm{relu}\big(\mathcal{W}_{w_2,b_2}(u_i^2)\big)$
7.     $(u_i^4, \Delta f_i) \leftarrow \mathcal{W}_{w_3,b_3}(u_i^3)$
8.     $s_i \leftarrow \mathrm{relu}(u_i^4)$
9.     $f_i \leftarrow f_{i-1} + \Delta f_i$
10. **end for**
11. $\mathcal{T}_{\boldsymbol{\Theta}}^\dagger(g) \leftarrow f_I$
```

(Codes: https://github.com/adler-j/learned_gradient_tomography
https://github.com/odlgroup/odl)

Two-dimensional computed tomography

The signal is real valued functions defined on a domain in $\mathbb{R}^2$ and X is a suitable vector space of such functions.
Forward operator is expressible in terms of the ray transform $\mathcal{P} : X \to Y$, and elements in Y are functions on lines

$$\mathcal{P}(f)(\ell) = \int_{\ell} f(x)dx \quad \text{for } \ell \in \mathbb{M}.$$

Non-linear forward operator:

$$\mathcal{T}(f)(\ell) = \lambda \exp\big(-\mu \mathcal{P}(f)(\ell)\big)$$

where $\lambda \in \mathbb{R}^+$ is the mean number of photons per pixel, and $\mu \in \mathbb{R}^+$ is the linear attenuation coefficient.

Two types of phantoms

- Ellipses: Randomly generated ellipses on a 128×128 pixel domain. The projection geometry was selected as a sparse 30 view parallel beam geometry with 5% additive Gaussian noise.
  Log-likelihood is $\mathcal{L}(\cdot, g) := \frac{1}{2}\|\cdot - g\|_Y^2$ which implies
  $$\nabla\big[\mathcal{L}(\mathcal{P}(\cdot), g)\big](f) = \mathcal{P}^*\big(\mathcal{P}(f) - g\big).$$
- Heads: Simulated projections of 512×512 pixel, 256×256 mm slices of CT scans of human heads. The acquisition geometry defining the data manifold was selected as a fan beam geometry with source-axis distance of 500 mm, source-detector distance 1000 mm, 1000 pixel, and 1000 angles. Poisson noise was added to the projections.
  Log-likelihood is
  $$\mathcal{L}(\mathcal{T}(f), g) := \int_{\mathbb{M}} \left(\mathcal{T}(f)(\ell) + g(\ell)\log\left(\frac{g(\ell)}{\mathcal{T}(f)(\ell)}\right)\right) d\ell$$
  which implies that
  $$\nabla\big[\mathcal{L}(\mathcal{T}(\cdot), g)\big](f) = -\mu\,\mathcal{P}^*\big(\mathcal{T}(f) - g\big).$$
- Regularizer is
  $$\mathcal{S}(f) := \frac{1}{2}\|\nabla f\|_2^2 \implies \nabla\mathcal{S}(f) = \nabla^*(\nabla f).$$

Implementation

- Initial guess: $\mathcal{T}^{\dagger}$ was given by the filtered back-projection (FBP) algorithm.
- Training-1: initially used 10^5 batches on the ellipses problem, where each batch contained 20 tomography problems. Learning rate starting at 10^{-3} and decayed down to about 10^{-5}.
  This training took four days on a workstation with a single Nvidia GTX Titan GPU.
- Training-2: These parameters were then used as an initial guess for heads problem. Learning rate starting at 10^{-5} and decreased to about 10^{-7}, each batch containing only one tomographic problems.
  This training took four days.

Comparison with standard methods

Table 1. Comparison of the learned method with standard methods.

	PSNR (dB)		Runtime (ms)	
Method	Ellipses	Heads	Ellipses	Heads
FBP	19.75	36.12	**4**	**130**
Learned	**32.02**	**43.82**	58	430
TV	29.83	38.40	11 963	173 845

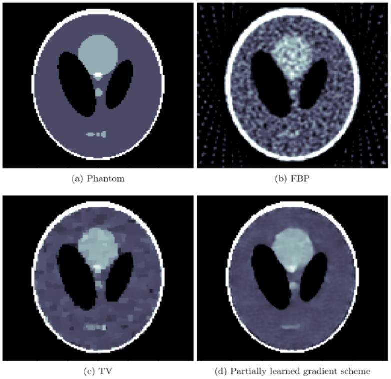


Figure 2. Reconstructing Shepp–Logan phantom using FBP, TV and the partially learned gradient scheme. Data is simulated from the Shepp–Logan phantom, which attains values between [0, 1]. All images are shown using a window set to [0.1, 0.4] for improved contrast.

Comparison with standard method

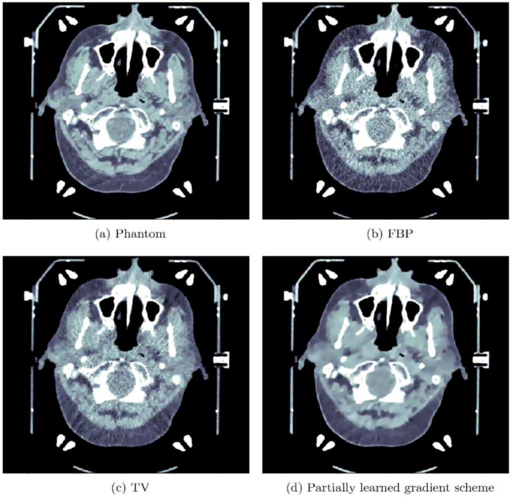


Figure 3. Reconstructing a head phantom using FBP, TV and the partially learned gradient scheme. Data is simulated from a physiological head phantom, which includes some weak streaks (so these are part of the ground truth). All images are shown using a window set to $[-200, 200]$ HU.

Iterates of the partially learned gradient scheme

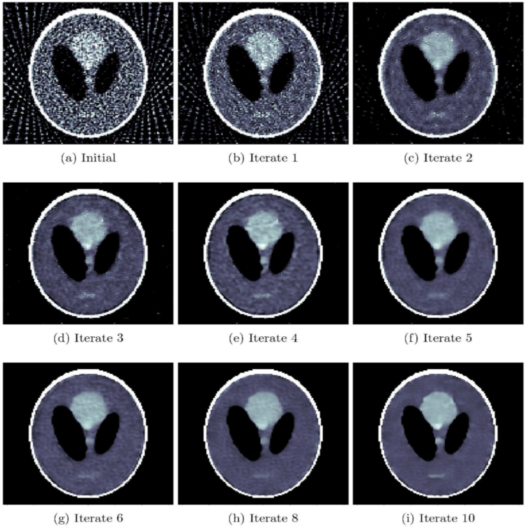


Figure 4. Iterates of the partially learned gradient scheme when applied to reconstruct the Shepp–Logan phantom. The initial iterate is given by the FBP method.

Iterates of the partially learned gradient scheme

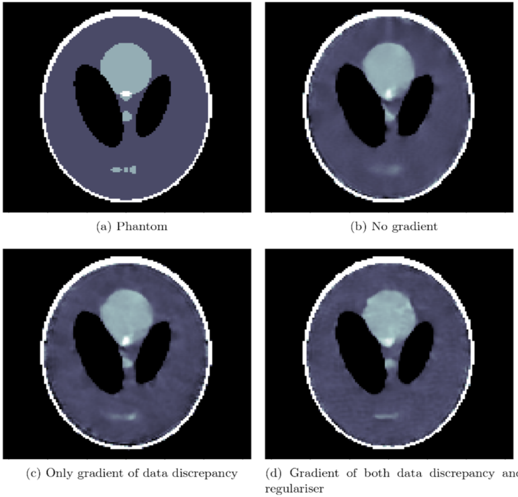

(a) Phantom
(b) No gradient
(c) Only gradient of data discrepancy
(d) Gradient of both data discrepancy and regulariser

Figure 5. Comparison of reconstructions using the partially learned gradient scheme with and without the gradient information. Note that gradient of data discrepancy includes the derivative of the forward operator.

Theoretical issues

The scheme in Algorithm 3 is formulated in a functional analytic setting. The theory for DNN is not well established in the infinite dimensional setting and there are several open issues that remain to be answered.

- 1. What class of operators can be approximated by a given deep neural network?
- 2. Convergence of approximation, in what sense? to what accuracy?

Regularization properties of the partially learned pseudo-inverse $\mathcal{T}^{\dagger}_{\boldsymbol{\Theta}}$:

$$\left\| \mathcal{T}^{\dagger}_{\boldsymbol{\Theta}}\big(\mathcal{T}(f_{\text{true}}) + \delta g\big) - f^{*} \right\|_{X} \to 0 \quad \text{whenever} \quad \|\delta g\|_{Y} \to 0$$

for some parameter choice rule for the hyper-parameters and training data?

Extensions

- Regularizers that exploit some kind of sparsity using L_1-like norm $\Rightarrow$ non-differentiable objective functional.
- In tomographic imaging the reconstructed image serves as input for an image analysis part which involves complex procedures, like segmentation and object recognition, that currently require involvement of human expertise. There is a growing trend in including some of these into the inverse problem that is referred to as feature reconstruction.
  Perform feature reconstruction by adding a feature extraction network to the learned reconstruction scheme: composing $\mathcal{T}^{\dagger}_{\boldsymbol{\Theta}}$ with a feature extraction operator $\mathcal{R}\colon X \to \mathbb{F}$ to get $\mathcal{R} \circ \mathcal{T}^{\dagger}_{\boldsymbol{\Theta}}$.
- Choice of loss function: $\mathrm{E}(f) = \|f - f_{\text{true}}\|^2_X$ is perhaps not the best predictor of human observer performance on a given image.

Conclusion

- Partially learned approach for solving ill-posed inverse problems that can integrate prior knowledge about the inverse problem with learning from training data.
- Using prior knowledge about the forward operator, data acquisition, data noise model and regularizer can significantly improve the performance of deep learning based approaches.
- Next step: try to tackle fully three-dimensional tomographic problems while training on two-dimensional datasets.

Thank you!