Regularization by architecture: A deep prior approach for inverse problems

S. Dittmer, T.Kluth, P.Maaß, D. Otero Baguer

Center for Industrial Mathematics (ZeTeM)
University of Bremen

24.01.2019
Stockholm, Sweden

Paper:

https://export.arxiv.org/abs/1812.03889

Code:

https://github.com/otero-baguer/analytic-deep-prior

Outline

Section 1

Introduction

Preliminaries

Consider an operator $A : X \to Y$ between Hilbert spaces X and Y.

Inverse Problem (General task)

Given measured noisy data

$$y^\delta = Ax^\dagger + \tau, \tag{1}$$

obtain an approximation $\hat{x}$ for $x^\dagger$, where τ, with $\|\tau\| \leq \delta$, describes the noise in the measurement.

Preliminaries

Consider an operator $A : X \to Y$ between Hilbert spaces X and Y.

Inverse Problem (General task)

Given measured noisy data

$$y^\delta = Ax^\dagger + \tau, \tag{1}$$

obtain an approximation $\hat{x}$ for $x^\dagger$, where τ, with $\|\tau\| \leq \delta$, describes the noise in the measurement.

Preliminaries

Classical approach: Variational regularization

$$\hat{x}_\alpha = \arg\min \frac{1}{2}\|Ax - y^\delta\|^2 + \alpha\mathcal{R}(x) \tag{2}$$

Examples of hand-crafted priors:

- $\|x\|^2$
- $\|x\|_1$
- $TV(x)$

Remark: α selection

Preliminaries

Classical approach: Variational regularization

$$\hat{x}_\alpha = \arg\min \frac{1}{2}\|Ax - y^\delta\|^2 + \alpha\mathcal{R}(x) \tag{2}$$

Examples of hand-crafted priors:

- $\|x\|^2$
- $\|x\|_1$
- $TV(x)$

Remark: α selection

Preliminaries

Classical approach: Variational regularization

$$\hat{x}_\alpha = \arg\min \frac{1}{2}\|Ax - y^\delta\|^2 + \alpha\mathcal{R}(x) \tag{2}$$

Examples of hand-crafted priors:

- $\|x\|^2$
- $\|x\|_1$
- $TV(x)$

Remark: α selection

Deep learning and inverse problems

- Primal-Dual reconstructions
- Learned gradient descent
- Learned post-processing: $\mathcal{F}_\theta \circ A^\dagger$
- Learned regularizers: $\mathcal{R}_\theta$
- Learned priors and generative networks (GAN, VAE)

Drawbacks:

- Need a lot of data. How to get the ground-truths?
- Real data noise might be different from the one present on the training samples.

Deep learning and inverse problems

- Primal-Dual reconstructions
- Learned gradient descent
- Learned post-processing: $\mathcal{F}_\theta \circ A^\dagger$
- Learned regularizers: $\mathcal{R}_\theta$
- Learned priors and generative networks (GAN, VAE)

Drawbacks:

- Need a lot of data. How to get the ground-truths?
- Real data noise might be different from the one present on the training samples.

Is it possible to solve inverse problems using deep learning without any training data?

Generative Networks

Let's consider a generative Neural Network $\varphi_W(z)$ previously trained.

- W is fixed after the training phase.
- We can obtain images by sampling $\mathbf{z}$.

For solving inverse problems:

- $\hat{z} = \arg\min_z \|\varphi_W(z) - y^\delta\|$
- $\hat{x} = \varphi_W(\hat{z})$

Can we obtain images by sampling $\mathbf{W}$ for a fixed z using the same network architecture (without training)?

Generative Networks

Let's consider a generative Neural Network $\varphi_W(z)$ previously trained.

- W is fixed after the training phase.
- We can obtain images by sampling $\mathbf{z}$.

For solving inverse problems:

- $\hat{z} = \arg\min_z \|\varphi_W(z) - y^\delta\|$
- $\hat{x} = \varphi_W(\hat{z})$

Can we obtain images by sampling $\mathbf{W}$ for a fixed z using the same network architecture (without training)?

Generative Networks

Let's consider a generative Neural Network $\varphi_W(z)$ previously trained.

- W is fixed after the training phase.
- We can obtain images by sampling $\mathbf{z}$.

For solving inverse problems:

- $\hat{z} = \arg\min_z \|\varphi_W(z) - y^\delta\|$
- $\hat{x} = \varphi_W(\hat{z})$

Can we obtain images by sampling $\mathbf{W}$ for a fixed z using the same network architecture (without training)?

Section 2

Deep Image Prior (DIP)

Basic Idea[1]

Given measured noisy data

$$y^\delta = Ax^\dagger + \tau, \tag{3}$$

train a neural network $\varphi_W(z)$ with parameters W by minimizing the loss function

$$\|A\varphi_W(z) - y^\delta\|^2 \tag{4}$$

with respect to W, for a single fixed input z and output y^δ.

Then compute $\hat{x} = \varphi_W(z)$

[1] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. "Deep Image Prior". In: *CoRR* (2017). arXiv: 1711.10925.

Basic Idea[1]

Given measured noisy data

$$y^\delta = Ax^\dagger + \tau, \tag{3}$$

train a neural network $\varphi_W(z)$ with parameters W by minimizing the loss function

$$\|A\varphi_W(z) - y^\delta\|^2 \tag{4}$$

with respect to W, for a single fixed input z and output y^δ.

Then compute $\hat{x} = \varphi_W(z)$

[1] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. "Deep Image Prior". In: *CoRR* (2017). arXiv: 1711.10925.

Basic Idea[1]

Given measured noisy data

$$y^\delta = Ax^\dagger + \tau, \tag{3}$$

train a neural network $\varphi_W(z)$ with parameters W by minimizing the loss function

$$\|A\varphi_W(z) - y^\delta\|^2 \tag{4}$$

with respect to W, for a single fixed input z and output y^δ.

Then compute $\hat{x} = \varphi_W(z)$

[1] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. "Deep Image Prior". In: *CoRR* (2017). arXiv: 1711.10925.

Some insights

- The network $\varphi_W(z)$ has a standard U-Net-like architecture.
- It has enough expressive power to reproduce some noise.
- Optimization method with early stopping plays an important role.
- Solving each instance requires training the network.
- It takes a lot of time.

Some insights

- The network $\varphi_W(z)$ has a standard U-Net-like architecture.
- It has enough expressive power to reproduce some noise.
- Optimization method with early stopping plays an important role.
- Solving each instance requires training the network.
- It takes a lot of time.

Some insights

- The network $\varphi_W(z)$ has a standard U-Net-like architecture.
- It has enough expressive power to reproduce some noise.
- Optimization method with early stopping plays an important role.
- Solving each instance requires training the network.
- It takes a lot of time.

Some insights

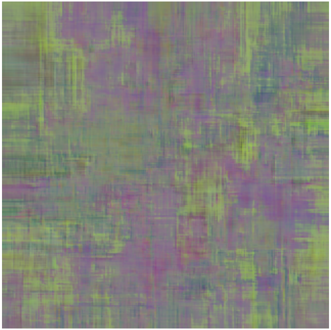
- The network $\varphi_W(z)$ has a standard U-Net-like architecture.
- It has enough expressive power to reproduce some noise.
- Optimization method with early stopping plays an important role.
- Solving each instance requires training the network.
- It takes a lot of time.

Some insights

- The network $\varphi_W(z)$ has a standard U-Net-like architecture.
- It has enough expressive power to reproduce some noise.
- Optimization method with early stopping plays an important role.
- Solving each instance requires training the network.
- It takes a lot of time.

Example

(a) Data (y^δ)



(b) Iteration 0

Example

(a) Data (y^δ)

(b) Iteration 50

Example

(a) Data (y^δ)

(b) Iteration 100

Example

(a) Data (y^δ)

(b) Iteration 150

Example

(a) Data (y^δ)

(b) Iteration 200

Example

(a) Data (y^δ)

(b) Iteration 250

Example

(a) Data (y^δ)

(b) Iteration 300

Example

(a) Data (y^δ)

(b) Iteration 350

Example

(a) Data (y^δ)

(b) Iteration 400

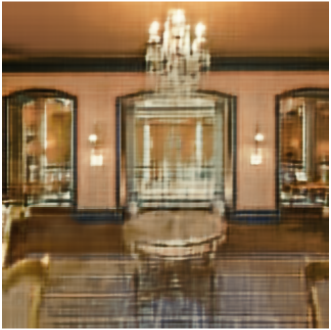
Example

(a) Data (y^δ)

(b) Iteration 450

Example

(a) Data (y^δ)



(b) Iteration 500

Example

(a) Data (y^δ)

(b) Iteration 550

Example

(a) Data (y^δ)

(b) Iteration 600

Example

(a) Data (y^δ)

(b) Iteration 650

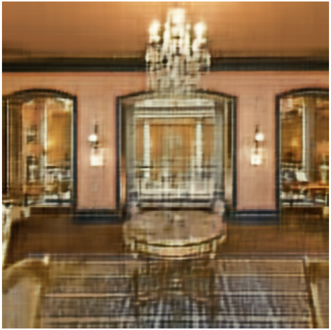
Example

(a) Data (y^δ)

(b) Iteration 700

Example

(a) Data (y^δ)



(b) Iteration 750

Example

(a) Data (y^δ)

(b) Iteration 800

Example

(a) Data (y^δ)

(b) Iteration 850

Example

(a) Data (y^δ)

(b) Iteration 900

Example

(a) Data (y^δ)

(b) Iteration 950

Example

(a) Data (y^δ)

(b) Iteration 1000

Example

(a) Data (y^δ)

(b) Iteration 1050

Example

(a) Data (y^δ)

(b) Iteration 1100

Example

(a) Data (y^δ)

(b) Iteration 1150

Example

(a) Data (y^δ)

(b) Iteration 1200

Example

(a) Data (y^δ)

(b) Iteration 1250

Example

(a) Data (y^δ)

(b) Iteration 1300

Example

(a) Data (y^δ)

(b) Iteration 1350

Example

(a) Data (y^δ)

(b) Iteration 1400

Example

(a) Data (y^δ)

(b) Iteration 1450

Example

(a) Data (y^δ)

(b) Iteration 1500

Example

(a) Data (y^δ)

(b) Iteration 1550

Example

(a) Data (y^{δ})

(b) Iteration 1600

Example

(a) Data (y^δ)

(b) Iteration 1650

Example

(a) Data (y^δ)

(b) Iteration 1700

Example

(a) Data (y^δ)

(b) Iteration 1750

Example

(a) Data (y^δ)

(b) Iteration 1800

Example

(a) Data (y^δ)

(b) Iteration 1850

Example

(a) Data (y^δ)

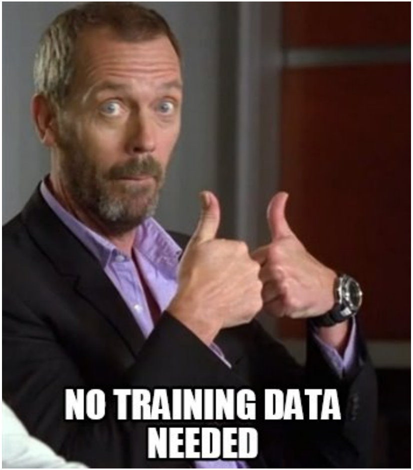
(b) Iteration 1900

DIP vs Global-Local GAN[2]

(a) Iteration 1900

(b) Global-Local GAN

[2] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. "Globally and Locally Consistent Image Completion". In: *ACM Transactions on Graphics (Proc. of SIGGRAPH 2017)* 36.4 (2017).


NO TRAINING DATA
NEEDED

Section 3

Analytic Deep Prior

Trivial remark

Can the DIP approach be used to solve ill-posed inverse problems?

Consider a trivial network $\varphi_W(z) = W$, and that W corresponds to elements in X.

$\Longrightarrow$ The approximate solution to the inverse problem is given by $\hat{x} = \varphi_W(z) = W$.

$\Longrightarrow$ Minimizing $\|A\varphi_W(z) - y^\delta\|^2 = \|AW - y^\delta\|$ by gradient descent with respect to W is equivalent to the classical Landweber iteration.

$$\alpha \sim \frac{1}{n} \tag{5}$$

Trivial remark

Can the DIP approach be used to solve ill-posed inverse problems?

Consider a trivial network $\varphi_W(z) = W$, and that W corresponds to elements in X.

$\Longrightarrow$ The approximate solution to the inverse problem is given by $\hat{x} = \varphi_W(z) = W$.

$\Longrightarrow$ Minimizing $\|A\varphi_W(z) - y^\delta\|^2 = \|AW - y^\delta\|$ by gradient descent with respect to W is equivalent to the classical Landweber iteration.

$$\alpha \sim \frac{1}{n} \qquad (5)$$

Trivial remark

Can the DIP approach be used to solve ill-posed inverse problems?

Consider a trivial network $\varphi_W(z) = W$, and that W corresponds to elements in X.

$\implies$ The approximate solution to the inverse problem is given by $\hat{x} = \varphi_W(z) = W$.

$\implies$ Minimizing $\|A\varphi_W(z) - y^\delta\|^2 = \|AW - y^\delta\|$ by gradient descent with respect to W is equivalent to the classical Landweber iteration.

$$\alpha \sim \frac{1}{n} \tag{5}$$

Trivial remark

Can the DIP approach be used to solve ill-posed inverse problems?

Consider a trivial network $\varphi_W(z) = W$, and that W corresponds to elements in X.

$\implies$ The approximate solution to the inverse problem is given by $\hat{x} = \varphi_W(z) = W$.

$\implies$ Minimizing $\|A\varphi_W(z) - y^\delta\|^2 = \|AW - y^\delta\|$ by gradient descent with respect to W is equivalent to the classical Landweber iteration.

$$\alpha \sim \frac{1}{n} \tag{5}$$

Convex optimization reminder

In the variational approach we usually minimize:

$$J(x) = \frac{1}{2}\|Ax - y^\delta\|^2 + \alpha\mathcal{R}(x). \tag{6}$$

where $\mathcal{R}$ is convex but not differentiable.

The necessary first order condition for a minimizer is given by

$$0 \in A^*(Ax - y^\delta) + \alpha\partial\mathcal{R}(x) \tag{7}$$

$$x \in x + \lambda A^*(Ax - y^\delta) + \lambda\alpha\partial\mathcal{R}(x) \tag{8}$$

$$x - \lambda A^*(Ax - y^\delta) \in x + \lambda\alpha\partial\mathcal{R}(x). \tag{9}$$

which is equivalent to

$$\operatorname*{Prox}_{\lambda\alpha\mathcal{R}}\left(x - \lambda A^*(Ax - y^\delta)\right) = x. \tag{10}$$

Convex optimization reminder

In the variational approach we usually minimize:

$$J(x) = \frac{1}{2}\|Ax - y^\delta\|^2 + \alpha\mathcal{R}(x). \tag{6}$$

where $\mathcal{R}$ is convex but not differentiable.

The necessary first order condition for a minimizer is given by

$$0 \in A^*(Ax - y^\delta) + \alpha\partial\mathcal{R}(x) \tag{7}$$

$$x \in x + \lambda A^*(Ax - y^\delta) + \lambda\alpha\partial\mathcal{R}(x) \tag{8}$$

$$x - \lambda A^*(Ax - y^\delta) \in x + \lambda\alpha\partial\mathcal{R}(x). \tag{9}$$

which is equivalent to

$$\operatorname*{Prox}_{\lambda\alpha\mathcal{R}}\left(x - \lambda A^*(Ax - y^\delta)\right) = x. \tag{10}$$

Convex optimization reminder

In the variational approach we usually minimize:

$$J(x) = \frac{1}{2}\|Ax - y^\delta\|^2 + \alpha\mathcal{R}(x). \tag{6}$$

where $\mathcal{R}$ is convex but not differentiable.

The necessary first order condition for a minimizer is given by

$$0 \in A^*(Ax - y^\delta) + \alpha\partial\mathcal{R}(x) \tag{7}$$

$$x \in x + \lambda A^*(Ax - y^\delta) + \lambda\alpha\partial\mathcal{R}(x) \tag{8}$$

$$x - \lambda A^*(Ax - y^\delta) \in x + \lambda\alpha\partial\mathcal{R}(x). \tag{9}$$

which is equivalent to

$$\operatorname*{Prox}_{\lambda\alpha\mathcal{R}}\left(x - \lambda A^*(Ax - y^\delta)\right) = x. \tag{10}$$

Convex optimization reminder

Turning the fixed point condition into an iteration scheme yields

$$\begin{aligned} x^{k+1} &= \operatorname*{Prox}_{\lambda\alpha\mathcal{R}}\left(x^k - \lambda A^*(Ax^k - y^\delta)\right) && (11)\\ &= \operatorname*{Prox}_{\lambda\alpha\mathcal{R}}\left((I - \lambda A^*A)x^k + \lambda A^* y^\delta\right) \ . && (12) \end{aligned}$$

Rewriting $W = I - \lambda A^*A$, $b = \lambda A^*$ and $\phi(\cdot) = \mathrm{Prox}_{\lambda\alpha\mathcal{R}}(\cdot)$ yields

$$x^{k+1} = \phi\left(Wx^k + b\right) \tag{13}$$

Convex optimization reminder

Example

Consider $\mathcal{R}(x) = I_+(x)$ (indicator function for non-negative numbers)

$$\operatorname*{Prox}_{\lambda\alpha\mathcal{R}}(x) = \mathbf{ReLu}(x) \tag{14}$$

The iteration scheme $x^{k+1} = \phi\left(Wx^k + b\right)$ is quite similar to a Neural Network.

Convex optimization reminder

Example

Consider $\mathcal{R}(x) = I_+(x)$ (indicator function for non-negative numbers)

$$\operatorname*{Prox}_{\lambda\alpha\mathcal{R}}(x) = \mathbf{ReLu}(x) \tag{14}$$

The iteration scheme $x^{k+1} = \phi\left(Wx^k + b\right)$ is quite similar to a Neural Network.

Analytic Deep Prior

Now we consider the particular architecture of a fully connected feed-forward iterative network with L identical layers

$$\varphi_W(z) = x^L, \tag{15}$$

where

$$x^{k+1} = \phi\left(Wx^k + b\right) \tag{16}$$

for $k = 0, .., L-1$ and $x^0 = z$.

- ϕ is the proximal mapping of a regularizing functional $\lambda\alpha R$
- W is such that $I - W = \lambda B^* B$ for some B
- $b = \lambda B^* y^\delta$

Analytic Deep Prior

In this setting, $\varphi_W(z)$ is identical to the L-th iterate of the PG method for minimizing

$$J_B(x) = \frac{1}{2}\|Bx - y^\delta\|^2 + \alpha\mathcal{R}(x), \tag{17}$$

If $\varphi_W(z) = x(B) = \arg\min\ J_B(x)$: Updating W, i.e. B, changes the discrepancy term in the Tikhonov functional.

Definition

We call this setting an **analytic deep prior** if B is trained from a single data point y^δ by gradient descent applied to

$$\min_B \|Ax(B) - y^\delta\|^2. \tag{18}$$

Analytic Deep Prior

In this setting, $\varphi_W(z)$ is identical to the L-th iterate of the PG method for minimizing

$$J_B(x) = \frac{1}{2}\|Bx - y^\delta\|^2 + \alpha\mathcal{R}(x), \tag{17}$$

If $\varphi_W(z) = x(B) = \arg\min\ J_B(x)$: Updating W, i.e. B, changes the discrepancy term in the Tikhonov functional.

Definition

We call this setting an **analytic deep prior** if B is trained from a single data point y^δ by gradient descent applied to

$$\min_B \|Ax(B) - y^\delta\|^2. \tag{18}$$

Analytic Deep Prior

In this setting, $\varphi_W(z)$ is identical to the L-th iterate of the PG method for minimizing

$$J_B(x) = \frac{1}{2}\|Bx - y^\delta\|^2 + \alpha\mathcal{R}(x), \tag{17}$$

If $\varphi_W(z) = x(B) = \arg\min \; J_B(x)$: Updating W, i.e. B, changes the discrepancy term in the Tikhonov functional.

Definition

We call this setting an **analytic deep prior** if B is trained from a single data point y^δ by gradient descent applied to

$$\min_B \|Ax(B) - y^\delta\|^2. \tag{18}$$

Training/Optimization

The training of B for given data y^δ is achieved by a gradient descent method applied to

$$F(B) = \frac{1}{2}\|Ax(B) - y^\delta\|^2 \tag{19}$$

$$\text{s.t.} \quad x(B) = \arg\min_x \; J_B(x). \tag{20}$$

The stationary points are characterized by $\partial F(B) = 0$ and gradient descent iterations with stepsize η are given by

$$B^{\ell+1} = B^\ell - \eta \partial F(B^\ell). \tag{21}$$

Hence we need to compute the derivative of F with respect to B.

Training/Optimization

The training of B for given data y^δ is achieved by a gradient descent method applied to

$$F(B) = \frac{1}{2}\|Ax(B) - y^\delta\|^2 \tag{19}$$

$$\text{s.t.} \quad x(B) = \arg\min_x \; J_B(x). \tag{20}$$

The stationary points are characterized by $\partial F(B) = 0$ and gradient descent iterations with stepsize η are given by

$$B^{\ell+1} = B^\ell - \eta \partial F(B^\ell). \tag{21}$$

Hence we need to compute the derivative of F with respect to B.

Training/Optimization

The training of B for given data y^δ is achieved by a gradient descent method applied to

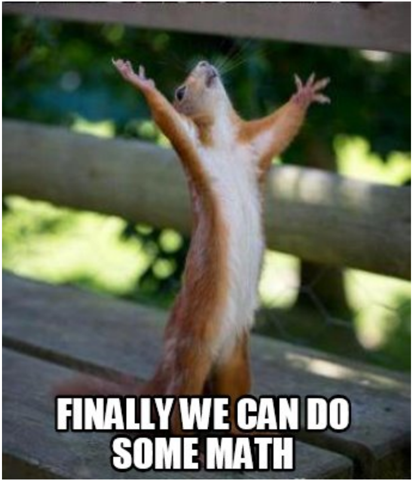
$$F(B) = \frac{1}{2}\|Ax(B) - y^\delta\|^2 \tag{19}$$

$$\text{s.t.} \quad x(B) = \arg\min_x \; J_B(x). \tag{20}$$

The stationary points are characterized by $\partial F(B) = 0$ and gradient descent iterations with stepsize η are given by

$$B^{\ell+1} = B^\ell - \eta \partial F(B^\ell). \tag{21}$$

Hence we need to compute the derivative of F with respect to B.


FINALLY WE CAN DO
SOME MATH

Example

Consider $\mathcal{R}(x) = \frac{1}{2}\|x\|^2$.

In this case $x(B) = \arg\min J_B(x) = (B^*B + \alpha I)^{-1}B^*y^\delta$.

For illustration we consider the rather unrealistic case $x^\dagger = u$, where u is a singular function of A ($Au = \sigma v$)

$$y^\delta = Au + \delta v = (\sigma + \delta)v \tag{22}$$

A lengthy computation exploiting $B^0 = A$ and $\beta_0 = \sigma$ shows that

$$B^{\ell+1} = B^\ell - c_\ell v u^* \tag{23}$$

Implementation

Goal: Find optimal B, to minimize the loss function

$$\frac{1}{2}\|Ax(B) - y^\delta\|^2 \tag{24}$$

Equivalent to train the network $\varphi_W(z)$ for the single data point (z, y^δ) updating B by back-propagation.

How many layers should the network have in order to ensure that $\varphi_W(z) = x(B) = \arg\min J_B$?

Thousands of layers! (slow convergence of the PG method).

Prohibitive!

Implementation

Goal: Find optimal B, to minimize the loss function

$$\frac{1}{2}\|Ax(B) - y^\delta\|^2 \tag{24}$$

Equivalent to train the network $\varphi_W(z)$ for the single data point (z, y^δ) updating B by back-propagation.

How many layers should the network have in order to ensure that $\varphi_W(z) = x(B) = \arg\min J_B$?

Thousands of layers! (slow convergence of the PG method).

Prohibitive!

Implementation

Goal: Find optimal B, to minimize the loss function

$$\frac{1}{2}\|Ax(B) - y^\delta\|^2 \tag{24}$$

Equivalent to train the network $\varphi_W(z)$ for the single data point (z, y^δ) updating B by back-propagation.

How many layers should the network have in order to ensure that $\varphi_W(z) = x(B) = \arg\min J_B$?

Thousands of layers! (slow convergence of the PG method).

Prohibitive!

Implementation

Goal: Find optimal B, to minimize the loss function

$$\frac{1}{2}\|Ax(B) - y^\delta\|^2 \tag{24}$$

Equivalent to train the network $\varphi_W(z)$ for the single data point (z, y^δ) updating B by back-propagation.

How many layers should the network have in order to ensure that $\varphi_W(z) = x(B) = \arg\min J_B$?

Thousands of layers! (slow convergence of the PG method).

Prohibitive!

Implementation

Goal: Find optimal B, to minimize the loss function

$$\frac{1}{2}\|Ax(B) - y^\delta\|^2 \tag{24}$$

Equivalent to train the network $\varphi_W(z)$ for the single data point (z, y^δ) updating B by back-propagation.

How many layers should the network have in order to ensure that $\varphi_W(z) = x(B) = \arg\min J_B$?

Thousands of layers! (slow convergence of the PG method).

Prohibitive!

Implementation

Solution:

- Consider only a reduced network with a small number, $L = 10$, of layers
- Set the input to be the network's output after the previous iteration.

Figure: The implicit network with $(k+1)L$ layers. Here $\varphi^L_{W_k}$ refers to a block of L identical fully connected layers with weights $W_k = I - \lambda B_k^T B_k$ and $b_k = \lambda B_k^T y^\delta$.

Implementation

Solution:

- Consider only a reduced network with a small number, $L = 10$, of layers
- Set the input to be the network's output after the previous iteration.

$$z \to \underbrace{\boxed{\varphi^L_{W_0}} \to \boxed{\varphi^L_{W_1}} \to \boxed{\varphi^L_{W_2}} \to \cdots \to \boxed{\varphi^L_{W_k}}}_{k+1} \to x_k$$

Figure: The implicit network with $(k+1)L$ layers. Here $\varphi^L_{W_k}$ refers to a block of L identical fully connected layers with weights $W_k = I - \lambda B_k^T B_k$ and $b_k = \lambda B_k^T y^\delta$.

Section 4

Academic Example

Setup

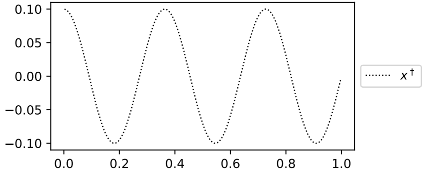
Consider the integration operator $A: \; L^2\left([0,1]\right) \; \rightarrow \; L^2\left([0,1]\right)$

$$(Ax)\left(t\right) = \int_0^t x(s)ds. \tag{25}$$

and

- $A_n \in \mathbb{R}^{n\times n}$: discretization of A.
- $x^\dagger \in \mathbb{R}^n$: one of the singular vectors u of A.
- $y^\delta = A_n x^\dagger + \tau$ with $\tau \sim \mathcal{N}(0, \sigma^2 \mathbb{1}_n)$

Ground-truth and data



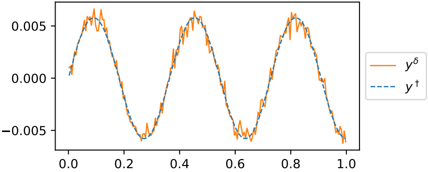
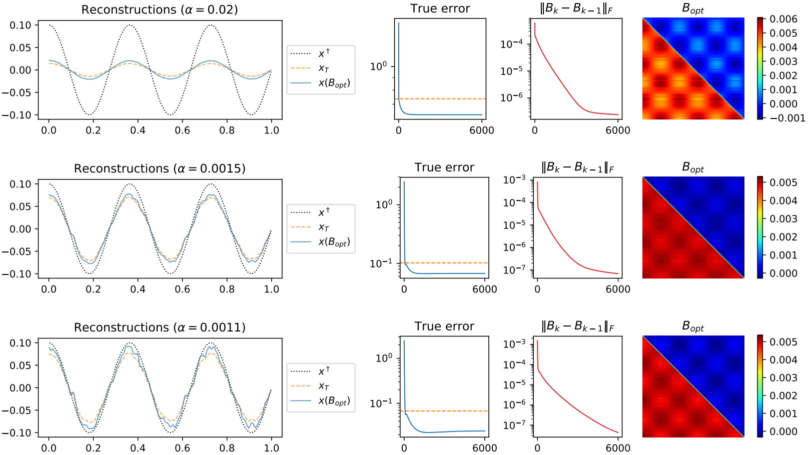


Figure: $x^\dagger = u_5$ and y^δ with $n = 200$ and 10% of noise.

Results $\left(R(\cdot) = \frac{1}{2}\|\cdot\|^2\right)$



Ground-truth and data

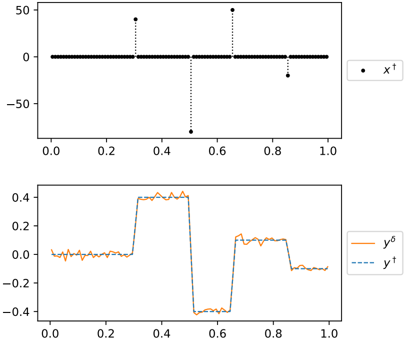
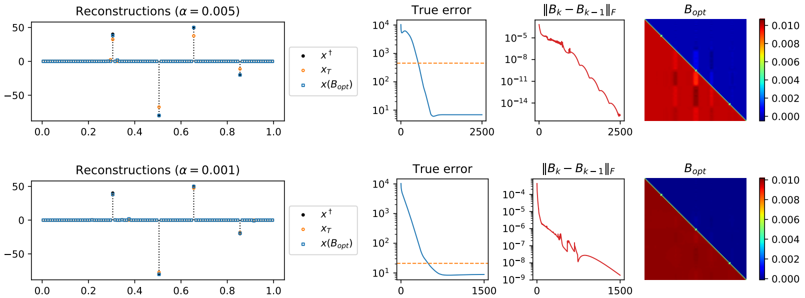


Figure: $x^\dagger$: sparse and y^δ with $n = 200$ and 10% of noise.

Results $(R(\cdot) = \|\cdot\|_1)$



Network convergence

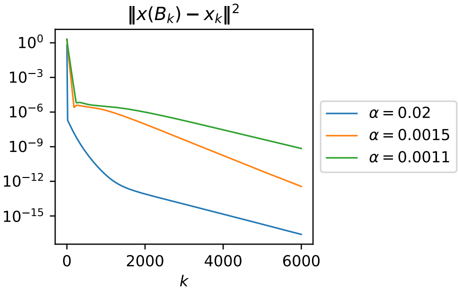
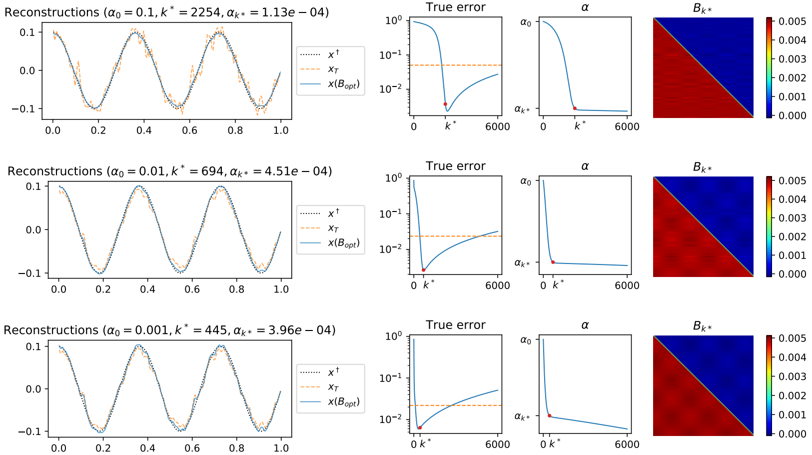


Figure: Difference between x_k and $x(B_k)$ after each training iteration k.

Results (adaptive α)



DFG Universität Bremen

Section 5

Magnetic Particle Imaging (MPI)

What is MPI?

Imaging modality based on injecting ferromagnetic nanoparticles which are consequently transported by the blood flow.

Goal: Measure the 3-D location and concentration of the nanoparticles.

Advantages:

- High spacial resolution (< 1mm)
- Measurement time (< 0.1 s)
- No harmful radiation

Figure: Magnetic particles developed in Lübeck

What is MPI?

Imaging modality based on injecting ferromagnetic nanoparticles which are consequently transported by the blood flow.

Goal: Measure the 3-D location and concentration of the nanoparticles.

Advantages:

- High spacial resolution (< 1mm)
- Measurement time (< 0.1 s)
- No harmful radiation

Figure: Magnetic particles developed in Lübeck

What is MPI?

Imaging modality based on injecting ferromagnetic nanoparticles which are consequently transported by the blood flow.

Goal: Measure the 3-D location and concentration of the nanoparticles.

Advantages:

- High spacial resolution (< 1mm)
- Measurement time (< 0.1 s)
- No harmful radiation

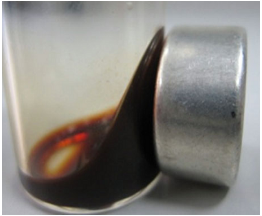


Figure: Magnetic particles developed in Lübeck

How it works?

- A magnetic field is applied, which is a superposition of:
  - static gradient field, which generates a field-free-point (FFP)
  - highly dynamic spatially homogeneous field, which moves the FFP in space.
- Mean magnetic moment of the nanoparticles in the neighborhood of the FFP generates an electro-magnetic field.
- Voltages are measured by so-called receive coils.
- The time-dependent measurements $v_\ell(t)$ in the receive coils constitute the data for reconstructing $c(x)$.

Inverse Problem

Linear Fredholm integral equation of the first kind describes the forward operator.

- Precisely modeling MPI is still an unsolved problem[3].
- The integral kernel is commonly determined in a time-consuming calibration procedure.

After discretization we end up with a linear system:

$$Sc = v \tag{26}$$

Goal: Reconstruct c from measured noisy data $v^\delta = Sc + \tau$.

[3] Tobias Kluth, Bangti Jin, and Guanglian Li. "On the degree of ill-posedness of multi-dimensional magnetic particle imaging". In: *Inverse Problems* 34.9 (2018).

Inverse Problem

Linear Fredholm integral equation of the first kind describes the forward operator.

- Precisely modeling MPI is still an unsolved problem[3].
- The integral kernel is commonly determined in a time-consuming calibration procedure.

After discretization we end up with a linear system:

$$Sc = v \tag{26}$$

Goal: Reconstruct c from measured noisy data $v^\delta = Sc + \tau$.

[3] Tobias Kluth, Bangti Jin, and Guanglian Li. "On the degree of ill-posedness of multi-dimensional magnetic particle imaging". In: *Inverse Problems* 34.9 (2018).

Inverse Problem

Linear Fredholm integral equation of the first kind describes the forward operator.

- Precisely modeling MPI is still an unsolved problem[3].
- The integral kernel is commonly determined in a time-consuming calibration procedure.

After discretization we end up with a linear system:

$$Sc = v \tag{26}$$

Goal: Reconstruct c from measured noisy data $v^\delta = Sc + \tau$.

[3] Tobias Kluth, Bangti Jin, and Guanglian Li. "On the degree of ill-posedness of multi-dimensional magnetic particle imaging". In: *Inverse Problems* 34.9 (2018).

Experimental setup

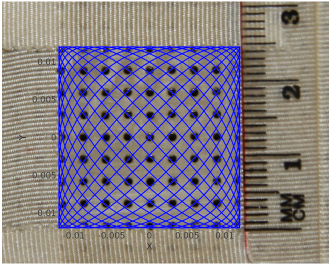
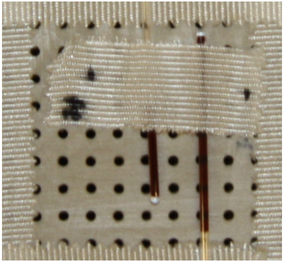


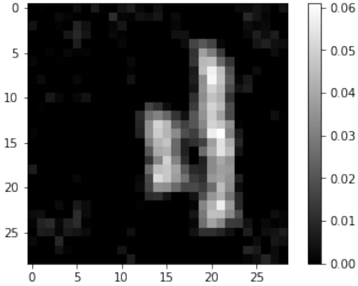
Figure: Used experimental platform with the FFP trajectory in blue. [4]

[4] Photo taken at University Medical Center Hamburg-Eppendorf by T. Kluth.

Results

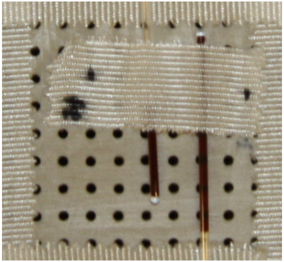


(a) Phantom (4mm)

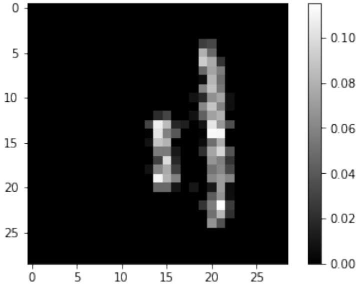


(b) Kacmarz reconstruction ($\alpha = 5 \cdot 10^{-4}$)

Results

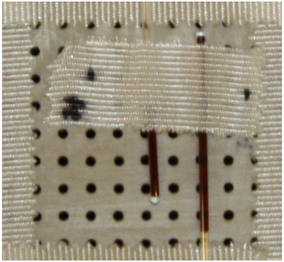


(a) Phantom (4mm)

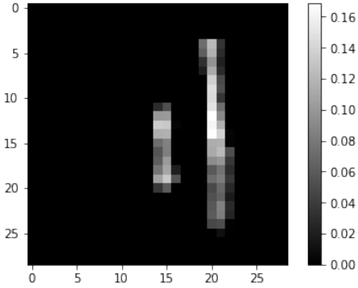


(b) l_1 reconstruction ($\alpha = 5 \cdot 10^{-3}$)

Results

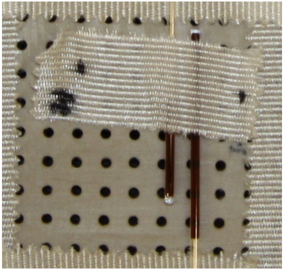


(a) Phantom (4mm)

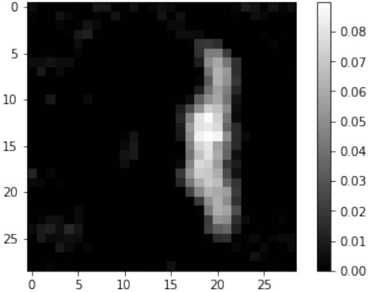


(b) DIP reconstruction ($lr = 5 \cdot 10^{-5}$)

Results

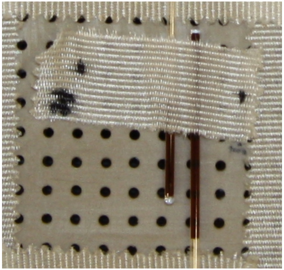


(a) Phantom (2mm)

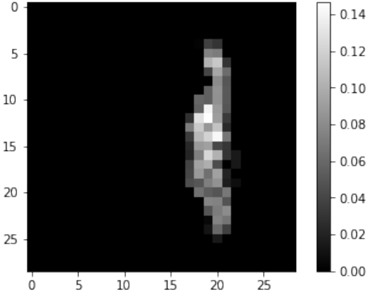


(b) Kacmarz reconstruction ($\alpha = 5 \cdot 10^{-4}$)

Results

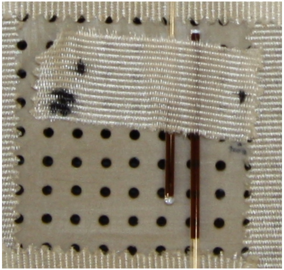


(a) Phantom (2mm)

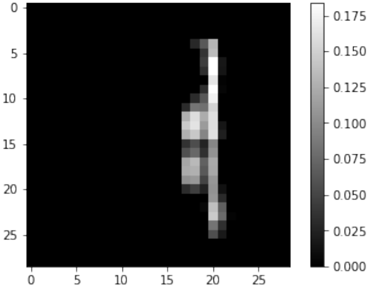


(b) l_1 reconstruction ($\alpha = 5 \cdot 10^{-3}$)

Results



(a) Phantom (2mm)



(b) DIP reconstruction ($lr = 5 \cdot 10^{-5}$)

Thanks!