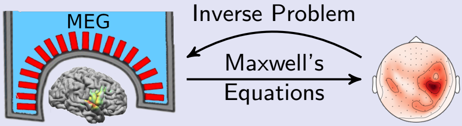
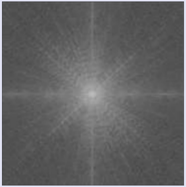
A Journey through Algorithm Unrolling for Inverse Problems

Thomas Moreau
INRIA Saclay - MIND Team

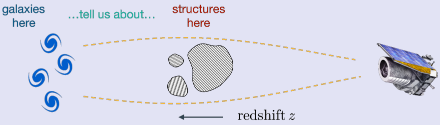
Neuroimaging – M/EEG



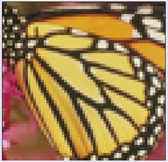
Neuroimaging – MRI



Astrophysics

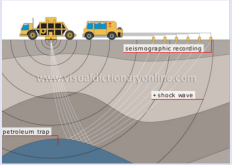


Imaging

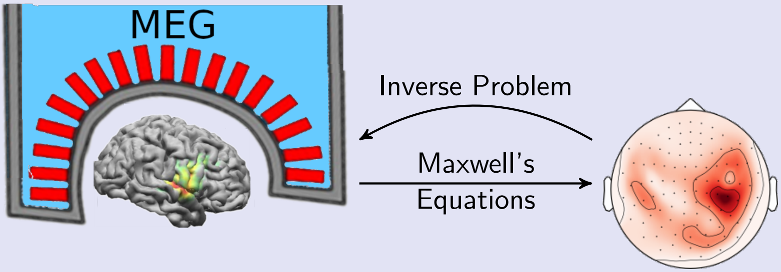


Super-Resolution, Inpainting, Deblurring, ...

Seismology – Prospection



Inverse Problem: Source Localization for M/EEG



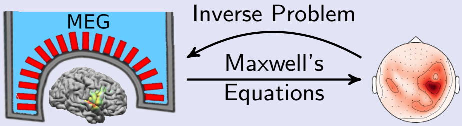
z — **Electrical activity** $\quad$ G $\quad$ x — **Observed signal**

Forward model: $x = Gz + \varepsilon$ $\quad$ **Inverse problem:** $z = f(x)$

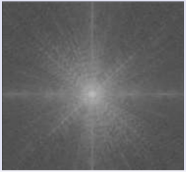
- Ill-posed problem: many solutions z such that $Gz = x$
- Noisy problem: need to account for ε

Inverse Problems

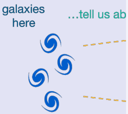
Neuroimaging – M/EEG



Neuroimaging – MRI

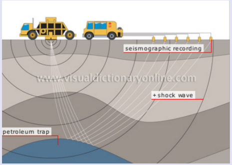


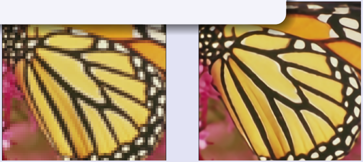
Astrophysics



For many inverse problems, we don't have access to large training sets with pairs (x, z).

Seismology – Prospection





Super-Resolution, Inpainting, Deblurring, ...

Inverse Problem Resolution

Regularized regression problem

$$f(x) = z^*(x) = \underset{z}{\operatorname{argmin}} \frac{1}{2}\|x - Gz\|_2^2 + \mathcal{R}(z)$$

where $\mathcal{R}$ encodes prior information to select a good/plausible solution.

Regularized regression problem

$$f(x) = z^*(x) = \operatorname*{argmin}_{z} \frac{1}{2}\|x - Gz\|_2^2 + \mathcal{R}(z)$$

where $\mathcal{R}$ encodes prior information to select a good/plausible solution.

Often solve this optimization problem many times for a given G,

Inverse Problem Resolution

Regularized regression problem

$$f(x) = z^*(x) = \operatorname*{argmin}_z \frac{1}{2}\|x - Gz\|_2^2 + \mathcal{R}(z)$$

where $\mathcal{R}$ encodes prior information to select a good/plausible solution.

Often solve this optimization problem many times for a given G,

$\Rightarrow$ Can we learn to solve such problem with unrolling?

$\Rightarrow$ With convergence guarantees toward the original solution $z^*(x)$?

$\neq$ *setting than supervised learning:* $\min_\Theta \mathbb{E}_{(x,z)} \frac{1}{2}\|z - \Phi_\Theta(x)\|_2^2$

Iterative Shrinkage-Thresholding Algorithm [Daubechies et al., 2004]

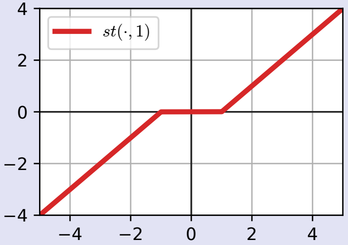
Proximal gradient descent algorithm with $\mathcal{R}(z) = \lambda\|z\|_1$,

$$z^{(t+1)} = st\left(z^{(t)} - \alpha \underbrace{\nabla f_x(z^{(t)})}_{G^\top(Gz^{(t)} - x)}, \alpha\lambda \right)$$

where α is a step size taken in $[0, \frac{2}{\|G\|_2^2}]$.

st is the soft-thresholding operator.

- Proximal operator for ℓ_1-norm.
- Push for sparse vector.



Iterative Shrinkage-Thresholding Algorithm [Daubechies et al., 2004]

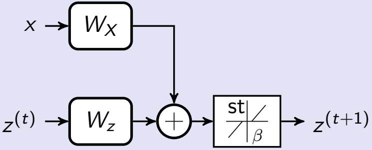
Proximal gradient descent algorithm with $\mathcal{R}(z) = \lambda \|z\|_1$,

$$z^{(t+1)} = st\left((Id - \alpha G^\top G) z^{(t)} + \alpha G^\top x, \alpha\lambda\right)$$

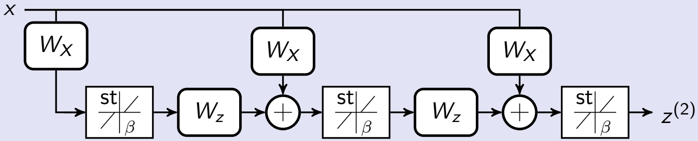
where α is a step size taken in $[0, \frac{2}{\|G\|_2^2}]$.

Computational graph interpretation:

- $W_z = Id - \alpha G^\top G$
- $W_X = \alpha G^\top$
- $\beta = \alpha\lambda$



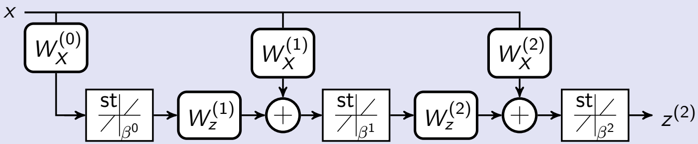
Unrolled ISTA:



Equivalent to ISTA with $W_z = Id - \alpha G^\top G$, $W_X = \alpha G^\top$ and $\beta = \alpha\lambda$.

3 iterations of ISTA $\Leftrightarrow$ 3 layers in the neural network

Learned ISTA:



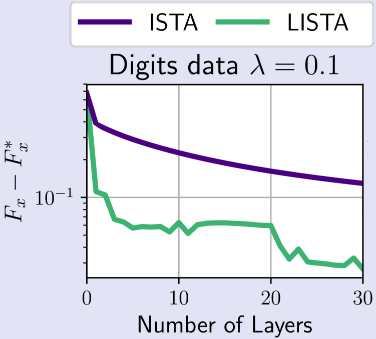
Learn $\Theta = (W_X^{(t)}, W_z^{(t)}, \beta^{(t)})_{t=0}^{T}$ s.t.

$$F_x(\Phi_\Theta(x)) \leq F_x(ISTA_T(x))$$

Goal:

$\Rightarrow$ Find the same solution as ISTA!

$\Rightarrow$ Faster?



Learning to optimize with unrolled ISTA

References

- **Moreau, T.** and Bruna, J. (2017). Understanding Neural Sparse Coding with Matrix Factorization.

  In *International Conference on Learning Representation (ICLR)*, Toulon, France

- Ablin, P., **Moreau, T.**, Massias, M., and Gramfort, A. (2019). Learning step sizes for unfolded sparse coding.

  In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 13100–13110, Vancouver, BC, Canada

How fast can LISTA go? [Moreau and Bruna, 2017]

Results are based on a quasi-diagonalization $G^\top G \simeq V^\top \Lambda V$ that does not distort "too much" the ℓ_1-norm.

- For a class of parameters, LISTA has the same cvg rate as ISTA.
- LISTA can benefit from improved constants.
- As the optimization approaches a solution, it is harder and harder to get improved constants.

$\Rightarrow$ Shows that it is possible to improve the first iterations of the algorithm.

ISTA: Majoration-Minimization

Taylor expansion of f_x in $z^{(t)}$

$$\begin{aligned} F_x(z) &= f_x(z^{(t)}) + \nabla f_x(z^{(t)})^\top (z - z^{(t)}) + \frac{1}{2}\|G(z - z^{(t)})\|_2^2 + \lambda\|z\|_1 \\ &\leq f_x(z^{(t)}) + \nabla f_x(z^{(t)})^\top (z - z^{(t)}) + \frac{L}{2}\|z - z^{(t)}\|_2^2 + \lambda\|z\|_1 \end{aligned}$$

$\Rightarrow$ Replace the Hessian $G^\top G$ by an upper bound L **Id**.

Separable function that can be minimized in close form

$$\begin{aligned} \underset{z}{\operatorname{argmin}} \frac{L}{2}\left\| z^{(t)} - \frac{1}{L}\nabla f_x(z^{(t)}) - z \right\|_2^2 + \lambda\|z\|_1 &= \operatorname{prox}_{\frac{\lambda}{L}}\left(z^{(t)} - \frac{1}{L}\nabla f_x(z^{(t)}) \right) \\ &= \operatorname{ST}\left(z^{(t)} - \frac{1}{L}\nabla f_x(z^{(t)}), \frac{\lambda}{L} \right) \end{aligned}$$

ISTA: Majoration-Minimization

Taylor expansion of f_x in $z^{(t)}$

$$\begin{aligned} F_x(z) &= f_x(z^{(t)}) + \nabla f_x(z^{(t)})^\top (z - z^{(t)}) + \frac{1}{2}\|G(z - z^{(t)})\|_2^2 + \lambda\|z\|_1 \\ &\leq f_x(z^{(t)}) + \nabla f_x(z^{(t)})^\top (z - z^{(t)}) + \frac{L}{2}\|z - z^{(t)}\|_2^2 + \lambda\|z\|_1 \end{aligned}$$

$\Rightarrow$ Replace the Hessian $G^\top G$ by an upper bound L **Id**.

By design,

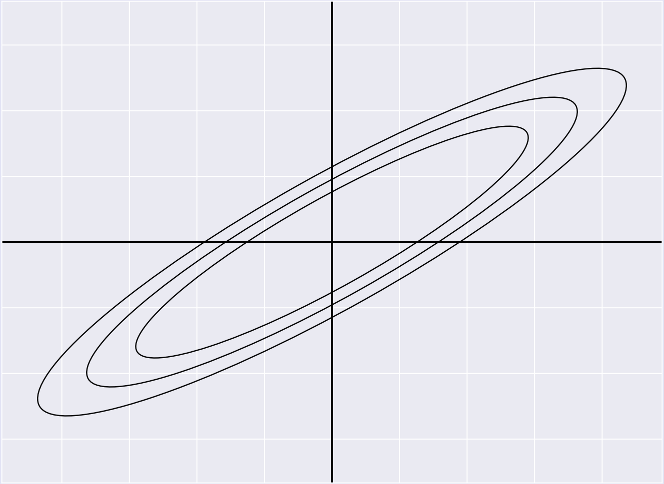
$$F(z^{t+1}) \leq Q^t(z^{t+1}) \leq Q^t(z^{t+1}) = F(z^t)$$

and the algorithm converges.

The key is to find a majorant easy to minimize.

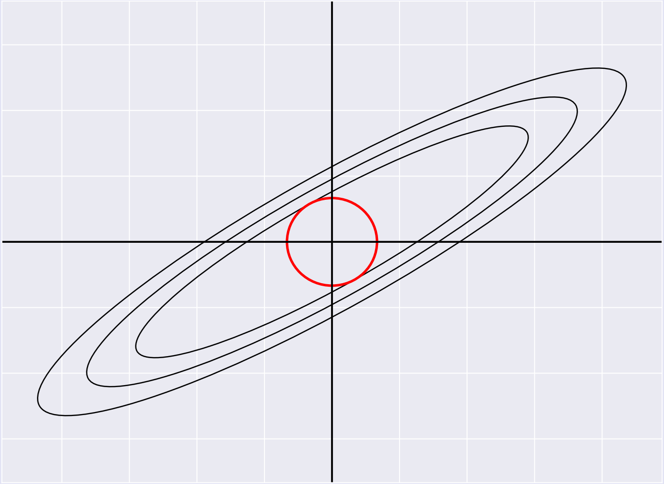
ISTA: Majoration for the data-fit

- Level sets for $z^\top G^\top G z$



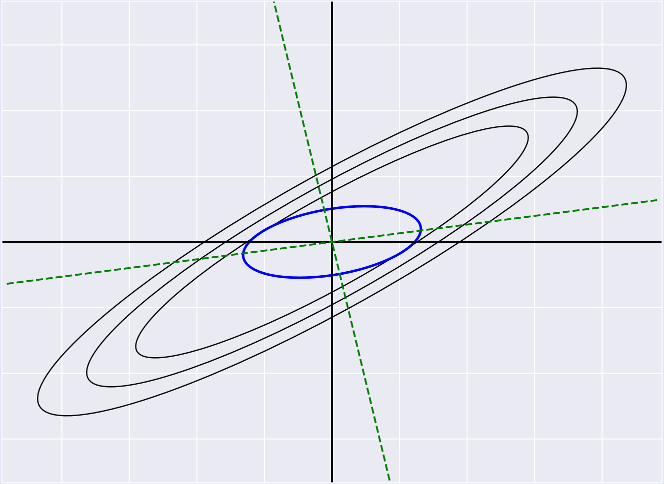
ISTA: Majoration for the data-fit

- Level sets for $z^\top G^\top G z \leq L\|z\|_2$



ISTA: Majoration for the data-fit

- Level sets for $z^\top G^\top G z \leq z^\top V^\top \Lambda V z$ [Moreau and Bruna, 2017]



What does LISTA learn? [Ablin et al., 2019]

Consider that the number of layers goes to $+\infty$.

Theorem – Asymptotic convergence of the weights

Assume that the weights of the network converge to a limit:

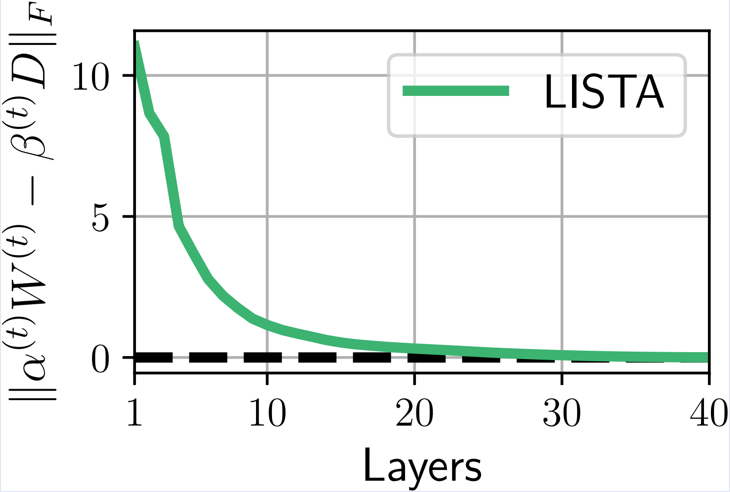
$$W_z^{(t)}, W_X^{(t)}, \beta^{(t)} \to W_z^*, W_X^*, \beta^* \qquad as \qquad t \to +\infty$$

and that the output of the network converges to a solution of the unsupervised problem.

Then

$$W_z^* = Id - \alpha D^\top D, \quad W_X^* = \alpha D^\top, \quad \beta^* = \alpha\lambda,$$

$\Rightarrow$ Correspond to ISTA with a learned step size α



40-layers LISTA network trained on a 10×20 problem with $\lambda = 0.1$

The weights $W^{(t)}$ align with D and α, β get coupled.

Step LISTA [Ablin et al., 2019]

Inspired by this result: learn adapted step sizes for ISTA.

Restricted parametrization : Only learn a step-size $\alpha^{(t)}$

$$z^{(t+1)} = \mathrm{ST}\left(z^{(t)} - \alpha^{(\mathbf{t})} D^\top (Dz^{(t)} - x), \lambda\alpha^{(\mathbf{t})}\right)$$

Fewer parameters:

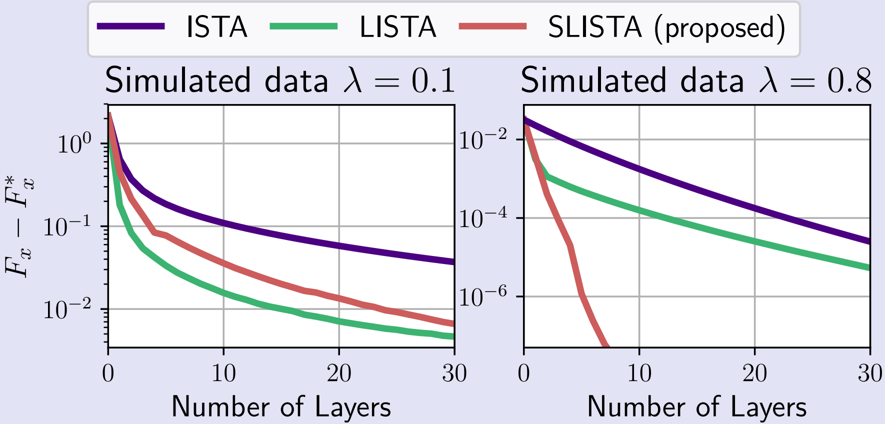
- Easier to learn
- Fewer degrees of freedom

$\Rightarrow$ Reduced performances?

Performances

Simulated data: $m = 256$ and $n = 64$

$$D_k \sim \mathcal{U}(\mathcal{S}^{n-1}) \text{ and } x = \frac{\widetilde{x}}{\|D^\top \widetilde{x}\|_\infty} \text{ with } \widetilde{x}_i \sim \mathcal{N}(0,1)$$

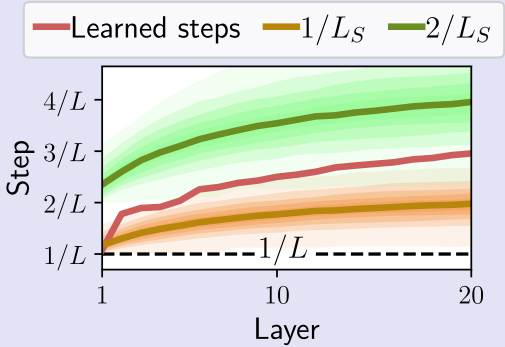


Learning better step sizes

Linked to SLISTA when step sizes are in $\left[\frac{1}{L_S}, \frac{2}{L_S}\right[$ when $Supp(z^{(t)}) = S$

L_S is the largest eigenvalue of $G^\top G$ restricted on the support S

$$\max_{\substack{Supp(z)=S \\ \|z\|_2 \leq 1}} zG^\top Gz$$



Unrolling for learned optimization

No hope to learn an algorithm that converges faster than ISTA uniformly.

- But one can learn parameters (step-size) of the algorithm that better adapt to the input distribution.

  [Ablin et al., 2019]

- Also possible to improve the first iterations of ISTA (improve constants).

  [Moreau and Bruna, 2017]

Also considered unrolled algorithms for TV in Cherkaoui, Sulam, **M.**, NeurIPS 2020.

A bilevel view on prior learning with unrolling

References

- Ablin, P., Peyré, G., and **Moreau, T.** (2020). Super-efficiency of automatic differentiation for functions defined as a minimum.

  In *International Conference on Machine Learning (ICML)*, volume 119, pages 32–41, Vienna, Austria (online). PMLR

- Malézieux, B., **Moreau, T.**, and Kowalski, M. (2022). Understanding approximate and Unrolled Dictionary Learning for Pattern Recovery.

  In *International Conference on Learning Representations (ICLR)*, online

Inverse Problem Prior: choosing $\mathcal{R}$.

Typical prior: Signal z is sparse in a specific dictionary D.

Synthesis formulation:
u sparse to synthesize $\boldsymbol{z} = Du$.

$$\min_{\substack{D,u \\ \|D_k\|_2 \leq 1}} \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{G} Du\|_2 + \lambda \|u\|_1 \quad .$$

Data driven dictionary: Learn D from the data $\boldsymbol{x}$.

[Olshausen and Field, 1997]

Bi-level formulation:

$$\min_{\|D_k\|\leq 1} h(D) \triangleq F(D, u^*(D)) \quad s.t. \quad u^*(D) = \operatorname*{argmin}_u F(D, u) \ .$$

Optimization problem in D solved with projected gradient descent.

$\Rightarrow$ How to estimate the gradient $g^*(D) = \nabla h(D)$ efficiently?

Unrolling for dictionary learning

Bi-level formulation:

$$\min_{\|D_k\|\leq 1} h(D) \triangleq F(D, u^*(D)) \quad s.t. \quad u^*(D) = \operatorname*{argmin}_u F(D, u) \ .$$

Optimization problem in D solved with projected gradient descent.

$\Rightarrow$ How to estimate the gradient $g^*(D) = \nabla h(D)$ efficiently?

Danskin Theorem: [Danskin, 1967]

$$g^*(D) = \nabla_1 F(D, u^*(D))$$

This is due to the fact that " $\nabla_2 F(D, u^*(D)) = 0$*".*

Unrolling for dictionary learning

Bi-level formulation:

$$\min_{\|D_k\|\leq 1} h(D) \triangleq F(D, u^*(D)) \quad s.t. \quad u^*(D) = \underset{u}{\operatorname{argmin}}\, F(D, u) \ .$$

Optimization problem in D solved with projected gradient descent.

$\Rightarrow$ How to estimate the gradient $g^*(D) = \nabla h(D)$ efficiently?

Danskin Theorem: [Danskin, 1967]

$$g^*(D) = \nabla_1 F(D, u^*(D))$$

This is due to the fact that "$\nabla_2 F(D, u^*(D)) = 0$".

Issue: computing $u^*(D)$ is computationally expansive.

Unrolling for dictionary learning

Unrolled formulation:

$$\min_{\|D_k\| \leq 1} h_T(D) \triangleq F(D, u_T(D)) \ .$$

The gradient estimate becomes:

$$g_T^2(D) = \nabla_1 F(D, u_T(D)) + J_T^\top \nabla_2 F(D, u_T(D))$$

Estimate the jacobian $J_T = \frac{\partial u_T}{\partial D}$ with back-propagation.

Unrolling for dictionary learning

Unrolled formulation:

$$\min_{\|D_k\|\leq 1} h_T(D) \triangleq F(D, u_T(D)) \ .$$

The gradient estimate becomes:

$$g_T^2(D) = \nabla_1 F(D, u_T(D)) + J_T^\top \nabla_2 F(D, u_T(D))$$

Estimate the jacobian $J_T = \frac{\partial u_T}{\partial D}$ with back-propagation.

Question: More efficient to use unrolling than classic AM?

- Work for smooth problems. [Ablin et al., ICML 2020]
- Improved performances for supervised learning. [Monga et al., 2021]

Alternate Minimization

No Jacobian estimation

$$g_T^1(D) = \nabla_1 F(D, u_T(D))$$

Unrolled ISTA

Account for Jacobian of u_T

$$\begin{aligned} g_T^2(D) = & \nabla_1 F(D, u_T(D)) \\ & + J_T^\top \nabla_2 F(D, u_T(D)) \end{aligned}$$

Alternate Minimization

No Jacobian estimation

$$g_T^1(D) = \nabla_1 F(D, u_T(D))$$

Converges as fast as u_T

$$\|g_T^1 - g^*\|_2 \leq L_1 \|u_T - u^*\|_2$$

Unrolled ISTA

Account for Jacobian of u_T

$$\begin{aligned} g_T^2(D) = & \nabla_1 F(D, u_T(D)) \\ & + J_T^\top \nabla_2 F(D, u_T(D)) \end{aligned}$$

Gradient Estimation

Alternate Minimization

No Jacobian estimation

$$g^1_T(D) = \nabla_1 F(D, u_T(D))$$

Converges as fast as u_T

$$\|g^1_T - g^*\|_2 \leq L_1 \|u_T - u^*\|_2$$

Unrolled ISTA

Account for Jacobian of u_T

$$\begin{aligned} g^2_T(D) =& \nabla_1 F(D, u_T(D)) \\ &+ J_T^\top \nabla_2 F(D, u_T(D)) \end{aligned}$$

May converge faster than u_T

$$\begin{aligned} \|g^2_T - g^*\| \leq& L \|J_T - J^*\|_2 \|u_T - u^*\|_2 \\ &+ L_2 \|u_T - u^*\|_2^2 \end{aligned}$$

$\Rightarrow$ Need to study $\|J_T - J^*\|_2$.

Differentiable unrolling of θ^t

Idea: Compute $J_T = \frac{\partial u_T}{\partial D}(D) \approx \frac{\partial u^*}{\partial D}(D)$ using automatic differentiation through an iterative algorithm.

Differentiable unrolling of θ^t

Idea: Compute $J_T = \frac{\partial u_T}{\partial D}(D) \approx \frac{\partial u^*}{\partial D}(D)$ using automatic differentiation through an iterative algorithm.

For the gradient descent algorithm:

$$u_{T+1} = u_T - \rho \frac{\partial G}{\partial z}(D, u_T)$$

The Jacobian reads,

$$\frac{\partial u_{T+1}}{\partial D}(D) = \left(Id - \rho \frac{\partial^2 G}{\partial z^2}(D, u_T) \right) \frac{\partial u_T}{\partial D}(D) - \rho \frac{\partial^2 G}{\partial z \partial D}(D, u_T)$$

Differentiable unrolling of θ^t

Idea: Compute $J_T = \frac{\partial u_T}{\partial D}(D) \approx \frac{\partial u^*}{\partial D}(D)$ using automatic differentiation through an iterative algorithm.

For the gradient descent algorithm:

$$u_{T+1} = u_T - \rho \frac{\partial G}{\partial z}(D, u_T)$$

The Jacobian reads,

$$\frac{\partial u_{T+1}}{\partial D}(D) = \left(Id - \rho \frac{\partial^2 G}{\partial z^2}(D, u_T) \right) \frac{\partial u_T}{\partial D}(D) - \rho \frac{\partial^2 G}{\partial z \partial D}(D, u_T)$$

$\Rightarrow$ Under smoothness conditions, if u_T converges to u^*, this converges toward $\frac{\partial u^*}{\partial D}(D)$

Context: min-min problems where $F = G$

$$\Rightarrow \text{Here, } \nabla_z F(D, u^*) = 0$$

Context: min-min problems where $F = G$

$$\Rightarrow \text{Here, } \nabla_z F(D, u^*) = 0$$

We consider the 3 gradient estimates:

- $g_1 = \nabla_D F(D, u_T)$ Analysis
- $g_2 = \nabla_D F(D, u_T) + \frac{\partial u_T}{\partial D}^\top \nabla_z F(D, u_T)$ Automatic
- $g_3 = \nabla_D F(D, u_T) - \frac{\partial^2 G}{\partial z \partial D}(D, u_T) \frac{\partial^2 G}{\partial z^2}^{-1}(D, u_T) \nabla_z F(D, u_T)$ Implicit

Context: min-min problems where $F = G$

$$\Rightarrow \text{Here, } \nabla_z F(D, u^*) = 0$$

We consider the 3 gradient estimates:

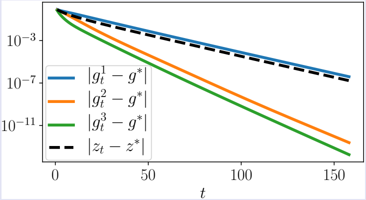
- $g_1 = \nabla_D F(D, u_T)$ Analysis
- $g_2 = \nabla_D F(D, u_T) + \frac{\partial u_T}{\partial D}^\top \nabla_z F(D, u_T)$ Automatic
- $g_3 = \nabla_D F(D, u_T) - \frac{\partial^2 G}{\partial z \partial D}(D, u_T) \frac{\partial^2 G}{\partial z^2}^{-1}(D, u_T) \nabla_z F(D, u_T)$ Implicit

Convergence rates: For G strongly convex in z,

$$|g_t^1(x) - g^*(x)| = O\left(|u_T(D) - u^*(D)|\right),$$

$$|g_t^2(x) - g^*(x)| = o\left(|u_T(D) - u^*(D)|\right),$$

$$|g_t^3(x) - g^*(x)| = O\left(|u_T(D) - u^*(D)|^2\right).$$



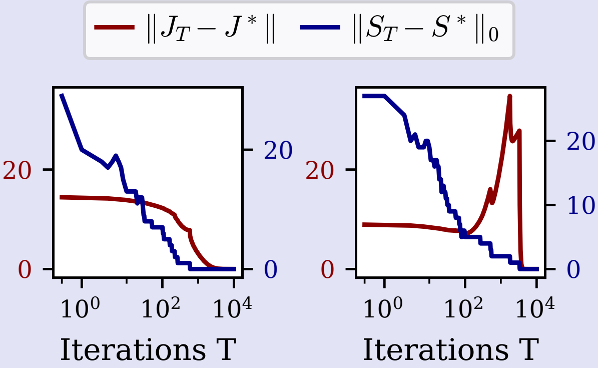
Convergence of the Jacobian

$$\|J_T - J^*\|_2 \leq A_T + B_T \ .$$

A_T converges linearly towards 0, B_T is an error term which may increase for large T and vanishes on the support of u^*.

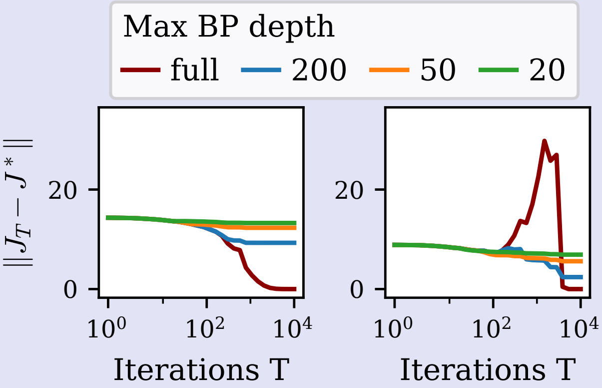
- On the support, the jacobian converges linearly.
- Before reaching the support, B_T is an error term that can accumulate.
- B_T can be attenuated with truncated back-propagation.

Empirical evaluation



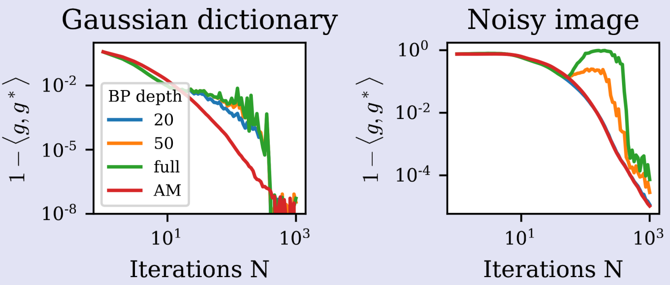
- Linear convergence once the support S^* is reached.
- Possible explosion before reaching S^*.

Empirical evaluation



- Truncated backpropagation (BP) reduces the explosion.
- Less precise when the support is reached.

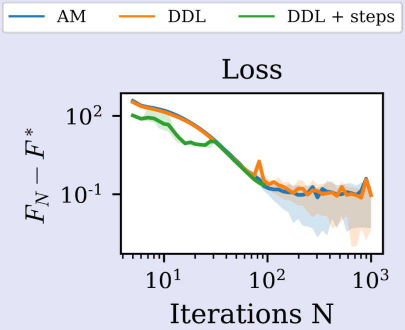
Numerical experiments on gradient



- **First iterations:** Stable behavior.
- **Too many iterations:** Numerical instabilities due to the accumulation of errors. Truncated back-propagation reduces the errors.
- **On the support:** Convergence towards g^*.

Comparison of 3 schemes to learn dictionaries on generated data:

- **AM:** use gradient estimate g^1_T
- **DDL:** use gradient estimate g^2_T
- **DDL+step:** DDL + learn the step size in the unrolled algorithm u_T.



$\Rightarrow$ Small number of iterations + learning step size improves uppon AM.

Unrolling for dictionary learning

Not the expected performance boost.

- Jacobian estimate stable only for a low number of iteration.
- Possible to design better dictionary learning algorithms but need extra ingredients.
- Maybe useful for task-driven dictionary learning.

We are currently investigating the interplay between G and the learning of D.

Thanks for your attention!

Slides are on my web page:

tommoral.github.io   @tomamoral