Learning divergences with unfolding: the case of phase retrieval in audio

Thomas Oberlin
Joint work with Pierre-Hugo Vial, Paul Magron & Cédric Févotte

ISAE-SUPAERO, IRIT, ANITI, Université de Toulouse

Workshop ASCETE

Grenoble, November 9th, 2023

Joint work with

Pierre-Hugo Vial
Now research engineer at Arturia, Grenoble

Paul Magron
Now research scientist at Inria Nancy

Cédric Févotte
Research director at CNRS, IRIT, Toulouse
Part of his ERC CoG FACTORY

Outline

1. Motivation : inverse problems in image and audio processing
2. Unfolded ADMM for Phase Retrieval
3. Results
4. Conclusion

Outline

Inverse problems in signal/image processing

Forward model of signal/image degradation :

$$\mathbf{y} = \mathbf{A}\mathbf{x}^* + \mathbf{n}, \tag{1}$$

Inverse problem (variational formulation)

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\varphi(\mathbf{x}) \tag{2}$$

Examples

- Denoising ($\mathbf{A} = \mathbf{I}$)
- Inpainting ($\mathbf{A}$ is a mask)
- Deblurring ($\mathbf{A}$ is a 2D convolution)
- Tomography ($\mathbf{A}$ computes radial projections)
- Compressed sensing ($\mathbf{A}$ satisfies conditions such as RIP)
- Super-resolution, pansharpening, dequantization, phase retrieval, etc

Neural networks for inverse problems

1. Learn the inverse operator f_θ such that, on a dataset $(\mathbf{x}_i, \mathbf{y}_i)$,

$$\mathbf{y}_i \approx f_\theta(\mathbf{x}_i).$$

   - Lack of stability and robustness
   - Black-box inversion, hard to interpret
   - No explicit use of the degradation model $\mathbf{A}$
   - Not generic : need for retraining for any change in model $\mathbf{A}$ or noise statistics

2. Learn the regularization only
   - Implicit, with denoisers (*plug-and-play*)
   - Explicit, with generative models

3. Learn the data-fitting divergence to use in place of $\|\cdot\|^2$

Unfolding/Unrolling

Unfolded neural networks

- Pick a splitting optimization scheme
- See it as a neural network
- Learn some parts of the scheme that become parameters of the network

Pros and cons

(+) Easy way to optimize hyperparameters / bilevel learning

(+) Interpretable architecture

(-) Unstabilities caused by autodiff

(-) What parameters should we learn ? What initialization ?

What parameters should we learn in unfolding ?

Most works focus on

- Hyper-parameters (eg, step size)
- Gradients or parts of them (linear operators, conv filters)
- Either free parameters or outputs of sub-networks

Proposal : learn the divergence

- Learn the proximal operator through parameterized activation functions
- Amounts to learning the divergence $D(\mathbf{y}|\mathbf{A}\mathbf{x})$

Unfolding in the literature

- Seminal paper : Gregor and Lecun, *Learning fast approximations of sparse coding*, ICML 2010
- A survey : Monga, Li and Eldar, *Algorithm Unrolling*, IEEE SP Mag., 2021
- Source separation : Hershey, Le Roux and Weninger, *Deep unfolding : Model-based inspiration of novel deep architectures*, 2014
- SISR : Wang et al., *Deep networks for image super-resolution with sparse prior*, ICCV 2015
- Dictionary learning : Malézieux, Moreau et Kowalski, *Understanding approximate and unrolled dictionary learning for pattern recovery*, ICLR 2022
- Others : Chouzenoux, Repetti, Pustelnik...

Outline

Phase retrieval in audio

$$\min_{\mathbf{x}\in\mathbb{R}^L} \| |\mathbf{A}\mathbf{x}|^d - \mathbf{r} \|^2, \tag{3}$$

- $\mathbf{A} \in \mathbb{C}^{K\times L}$ is the measurement operator
- $\mathbf{r} \in \mathbb{R}_+^K$ are the phaseless measurements
- $||.||$ denotes the Euclidean norm

In audio, $\mathbf{A}$ is often the short-time Fourier transform (STFT), $\mathbf{r}$ are either magnitude ($d = 1$) or power ($d = 2$) spectrograms

An ubiquitous problem

- Many audio processing pipelines operate on the spectrogram
- Need for the phase to go back to a waveform
- Examples : source separation, speech-to-speech, etc

But : why using ℓ_2^2 ?

Phase retrieval with Bregman divergences

Bregman divergence :

$$\mathcal{D}_\psi(\mathbf{p}\,|\,\mathbf{q}) = \sum_{k=1}^{K} \left[\psi(p_k) - \psi(q_k) - \psi'(q_k)(p_k - q_k)\right], \tag{4}$$

where $\mathbf{p}$, $\mathbf{q} \in \mathbb{R}^K$ and $\psi : \mathbb{R} \to \mathbb{R}$ is a strictly-convex, continuously-differentiable generating function (with derivative ψ').

Includes well-known divergences such as ℓ_2^2, Kullback-Leibler, Itakura-Saito...

PR with Bregman divergences

$$\min_{\mathbf{x}\in\mathbb{R}^L} \mathcal{D}_\psi(\mathbf{r}\,|\,|\mathbf{A}\mathbf{x}|^d) \text{ right formulation,} \tag{5}$$

$$\min_{\mathbf{x}\in\mathbb{R}^L} \mathcal{D}_\psi(|\mathbf{A}\mathbf{x}|^d\,|\,\mathbf{r}) \text{ left formulation.} \tag{6}$$

(7)

Special cases of Bregman divergences

Divergence	$d_\psi(y\|z)$	$\psi(z)$	Bruit
Quadratic loss	$\frac{1}{2}(y-z)^2$	$\frac{1}{2}z^2$	Gaussien
Kullback-Leibler	$y(\log y - \log z) - (y-z)$	$z \log z$	Poisson
Itakura-Saito	$\frac{y}{z} - \log \frac{y}{z} - 1$	$-\log z$	Gamma mult
beta-divergence	$\frac{y^\beta}{\beta-1} - \frac{\beta y z^{\beta-1}}{\beta-1} + z^\beta$	$\frac{z^\beta}{\beta(\beta-1)} - \frac{z}{\beta-1} + \frac{1}{\beta}$	

Gradient descent

Gradient computation

$$\nabla J(\mathbf{x}) = \frac{d}{2}\,\mathbf{A}^{\mathsf{H}}\left[|\mathbf{A}\mathbf{x}|^{d-2} \odot (\mathbf{A}\mathbf{x}) \odot \mathbf{g}_{\psi}\right]. \tag{8}$$

$$\text{with } \mathbf{g}_{\psi} = \psi''(|\mathbf{A}\mathbf{x}|^{d}) \odot (|\mathbf{A}\mathbf{x}|^{d} - \mathbf{r}) \qquad \text{for "right" PR,} \tag{9}$$

$$\mathbf{g}_{\psi} = \psi'(|\mathbf{A}\mathbf{x}|^{d}) - \psi'(\mathbf{r}) \qquad \text{for "left" PR.} \tag{10}$$

Gradient descent

- Standard :

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \mu \nabla J(\mathbf{x}_t). \tag{11}$$

- With Nesterov-like acceleration :

$$\begin{aligned} \mathbf{q}_{t+1} &= \mathbf{x}_t - \mu \nabla J(\mathbf{x}_t), \\ \mathbf{x}_{t+1} &= \mathbf{q}_{t+1} + \eta(\mathbf{q}_{t+1} - \mathbf{q}_t), \end{aligned} \tag{12}$$

Special cases for ℓ_2^2 : Griffin-Lim algorithms GLA [Griffin and Lim, 1984] and FGLA [Perraudin et al., 2013] ; Wirtinger flow [Candès et al. 2013]

ADMM

Constrained formulation

$$\min_{\mathbf{x}\in\mathbb{R}^L,\mathbf{u}\in\mathbb{R}_+^K,\theta\in[0;2\pi[^K} \mathcal{D}_\psi(\mathbf{u}\,|\,\mathbf{r}) \quad \text{s.t.} \quad (\mathbf{A}\mathbf{x})^d = \mathbf{u}\odot e^{i\boldsymbol{\theta}}, \tag{13}$$

ADMM

$$\mathbf{h}_{t+1} = (\mathbf{A}\mathbf{x}_t)^d + \frac{\boldsymbol{\lambda}_t}{\rho} \tag{14}$$

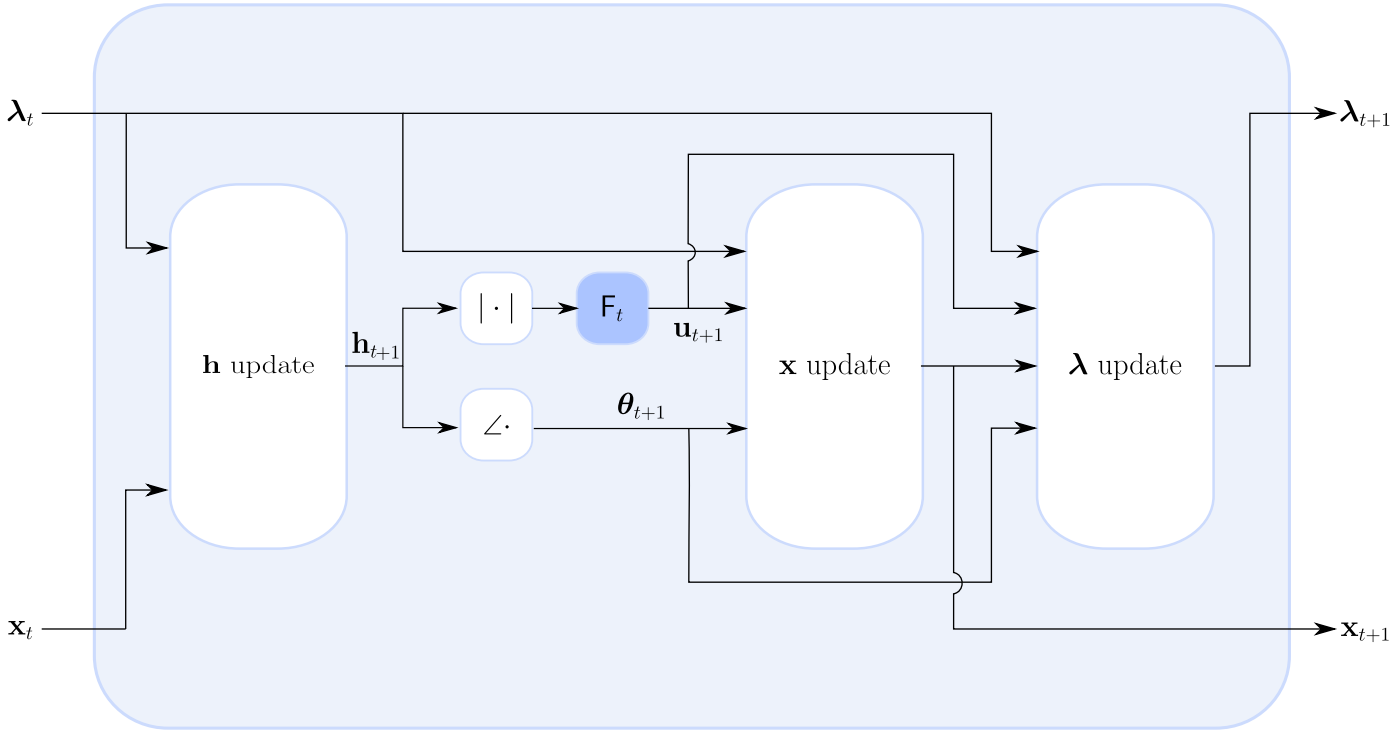
$$\mathbf{u}_{t+1} = \text{prox}_{\rho^{-1}\mathcal{D}_\psi(\cdot\,|\,\mathbf{r})}(|\mathbf{h}_{t+1}|) \tag{15}$$

$$\boldsymbol{\theta}_{t+1} = \angle\mathbf{h}_{t+1} \tag{16}$$

$$\mathbf{x}_{t+1} = \mathbf{A}^{\mathsf{H}}\Big(\mathbf{u}_{t+1}\odot e^{i\boldsymbol{\theta}_{t+1}} - \frac{\boldsymbol{\lambda}_t}{\rho}\Big)^{1/d} \tag{17}$$

$$\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \rho(\mathbf{A}\mathbf{x}_{t+1} - \mathbf{u}_{t+1}\odot e^{i\boldsymbol{\theta}_{t+1}}), \tag{18}$$

Unfolding ADMM phase retrieval



One layer of the proposed unfolded architecture

Unfolding ADMM phase retrieval

1 iteration = 1 layer of a neural net :

$$(\mathbf{x}_T,\, \boldsymbol{\lambda}_T) = \mathbf{U}(\mathbf{x}_0,\, \boldsymbol{\lambda}_0) = \mathsf{U}_1 \circ \cdots \circ \mathsf{U}_T(\mathbf{x}_0,\, \boldsymbol{\lambda}_0), \tag{19}$$

where t-th layer U_t can be decomposed into two linear parts denoted by $\mathsf{L}_t^{(1)}$ and $\mathsf{L}_t^{(2)}$, and a nonlinear part NL_t as follows :

$$\mathsf{L}_t^{(1)} : (\mathbf{x}_{t-1},\, \boldsymbol{\lambda}_{t-1}) \mapsto \mathbf{h}_t \tag{20}$$

$$\mathsf{NL}_t : \mathbf{h}_t \mapsto (\mathbf{u}_t,\, \boldsymbol{\theta}_t) = (\mathsf{F}_t(|\mathbf{h}_t|,\, \mathbf{r}),\, \angle \mathbf{h}_t) \tag{21}$$

$$\mathsf{L}_t^{(2)} : (\mathbf{x}_{t-1},\, \boldsymbol{\lambda}_{t-1}, \mathbf{u}_t,\, \boldsymbol{\theta}_t) \mapsto (\mathbf{x}_t,\, \boldsymbol{\lambda}_t). \tag{22}$$

In ADMM, F_t is defined as

$$\mathsf{F}_t(y, r) = \mathrm{prox}_{\rho^{-1}\mathcal{D}_\psi(\cdot\,|\,\mathbf{r})}(\mathbf{y}) = \mathrm{prox}_{\rho^{-1}\tilde{\psi}}(\mathbf{y} + \rho^{-1}\psi'(\mathbf{r})). \tag{23}$$

where $\tilde{\psi}(\mathbf{z}) = \sum_k \psi(z_k)$.

Learning the proximal operator

- Recall that in the ADMM scheme,

$$\mathsf{F}_t(\mathbf{y}, \mathbf{r}) = \mathrm{prox}_{\rho^{-1}\tilde{\psi}}(\mathbf{y} + \rho^{-1}\psi'(\mathbf{r})). \tag{24}$$

- Proposal : replace the prox by a learnable activation function defined by :

$$\mathrm{APL}(\mathbf{y}) := \max(\mathbf{y}, 0) + \sum_{c=1}^{C} w_c \max(-\mathbf{y} + b_c, 0), \tag{25}$$

- Reparameterization to account for $\mathbf{r}$

$$\mathsf{F}_t(\mathbf{y}, \mathbf{r}) = \mathrm{APL}_t\left(\gamma_t^{(1)}\mathbf{y} + \gamma_t^{(2)}\frac{\mathbf{r}^{\beta_t - 1}}{\beta_t - 1}\right), \tag{26}$$

  with learnable parameters $w_{c,t}$, $b_{c,t}$, $\gamma_t^{(1)}$, $\gamma_t^{(2)}$, and β_t.
- Two variants, **tied** and **untied**

From APL to divergence learning

Proposition

Under mild conditions, there exists a function $f_{\mathbf{r},t} : \mathbb{R}^K \to \mathbb{R} \cup \{+\infty\}$ such that $\mathrm{F}_t(\mathbf{y}, \mathbf{r}) = \mathrm{prox}_{f_{\mathbf{r},t}}(\mathbf{y})$

In the **tied** variant, we know exactly what we optimize

Closed-form expression :

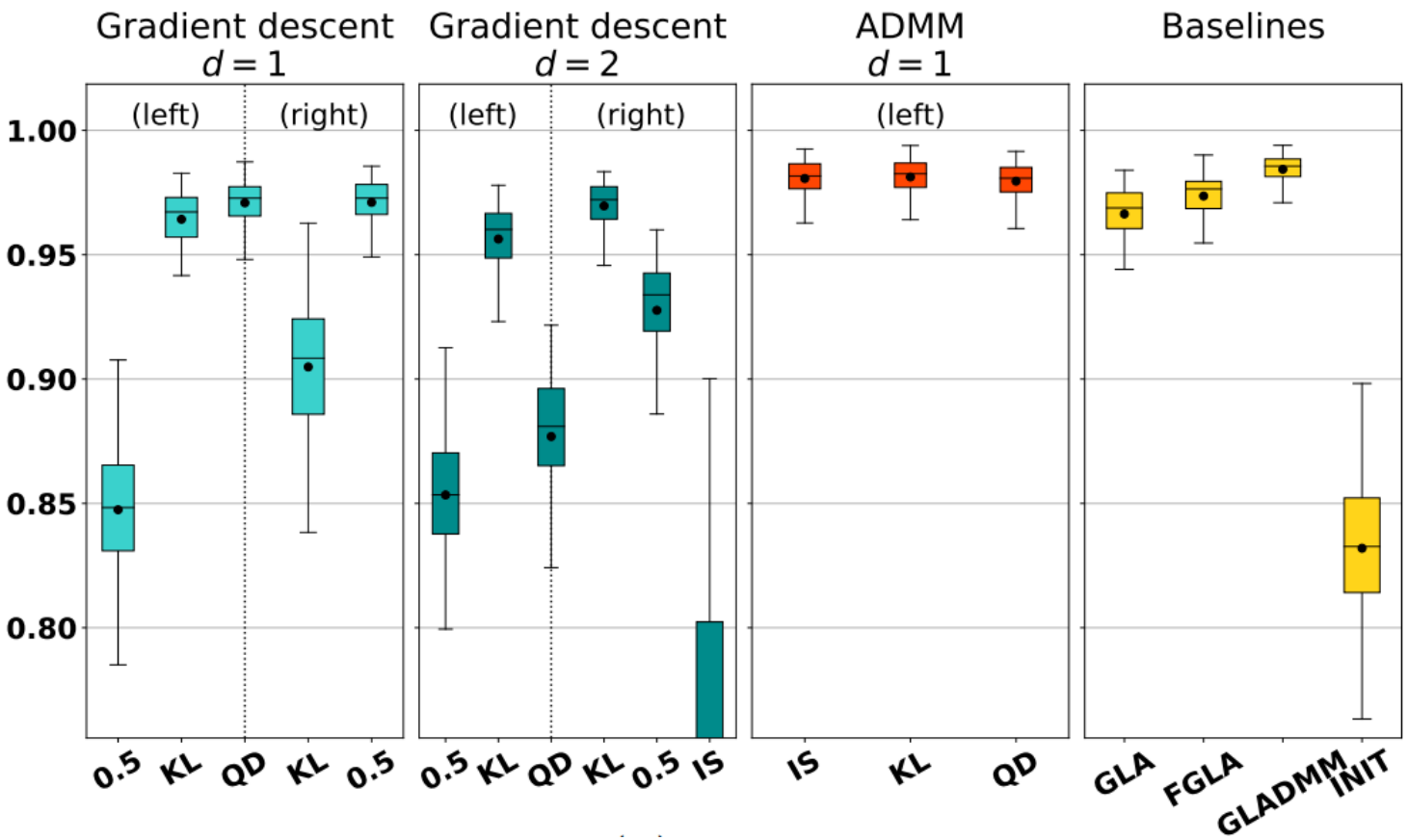
$$f_{\mathbf{r}}(\mathbf{y}) = \frac{1}{\gamma^{(1)}} \left\langle \mathrm{APL}^{-1}(\mathbf{y}) - \gamma^{(2)} \frac{\mathbf{r}^{\beta-1}}{\beta-1}, \mathbf{y} \right\rangle - \frac{1}{2}\|\mathbf{y}\|^2 - \frac{1}{\gamma^{(1)}} \widetilde{\mathrm{APL}}(\mathrm{APL}^{-1}(\mathbf{y})). \quad (27)$$

Outline

Experimental setting

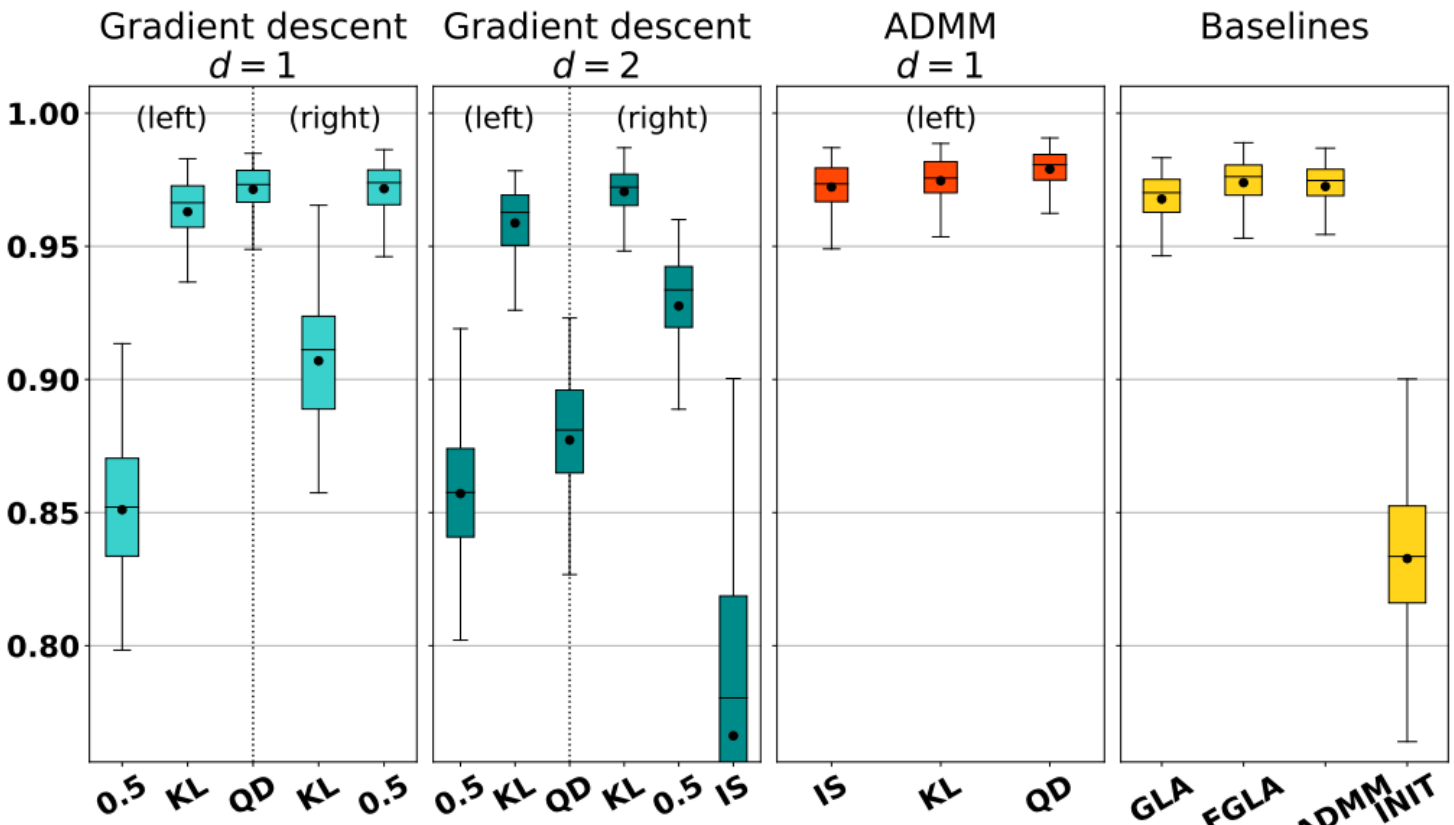
- Data : 2-second signals from TIMIT dataset. 1000 for Train, 10 for val, 50 for test
- STFT with 1024 samples-long (46 ms) self-dual sine window
- Parameters : $T = 15$ layers and $C = 3$ pieces in APLs
- Training : Adam optimizer, neg-STOI loss
- Metric : STOI $\in [0, 1]$, the higher the better
- Two versions : **tied** and **untied**
- Baselines : GLA, ADMM ℓ_2^2

GLA vs ADMM



Comparisons between GD and ADMM for several divergences, exact spectrograms

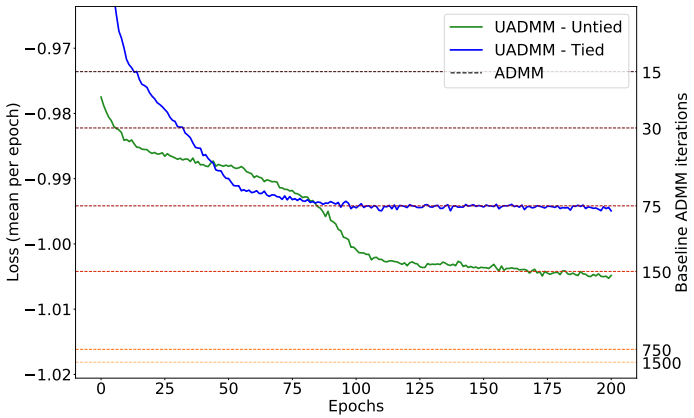
GLA vs ADMM



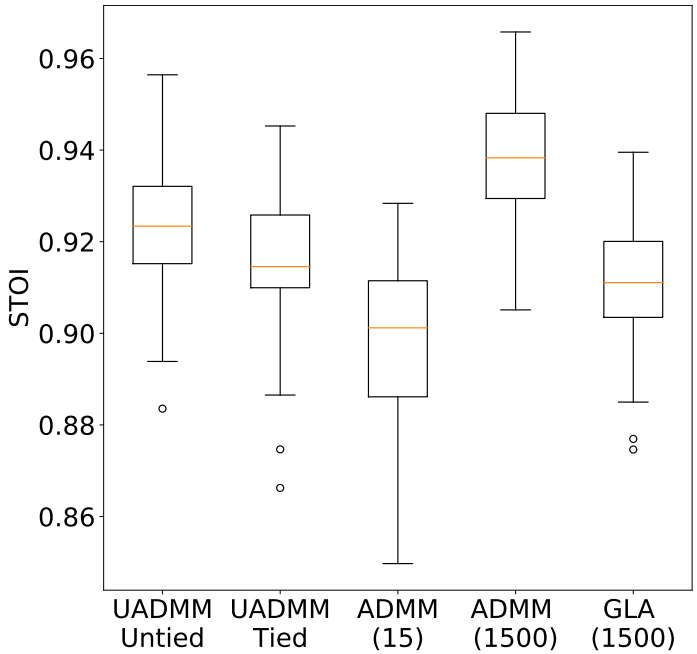
(a) +10 dB

Comparisons between GD and ADMM for several divergences, modified spectrograms

Impact of divergence learning

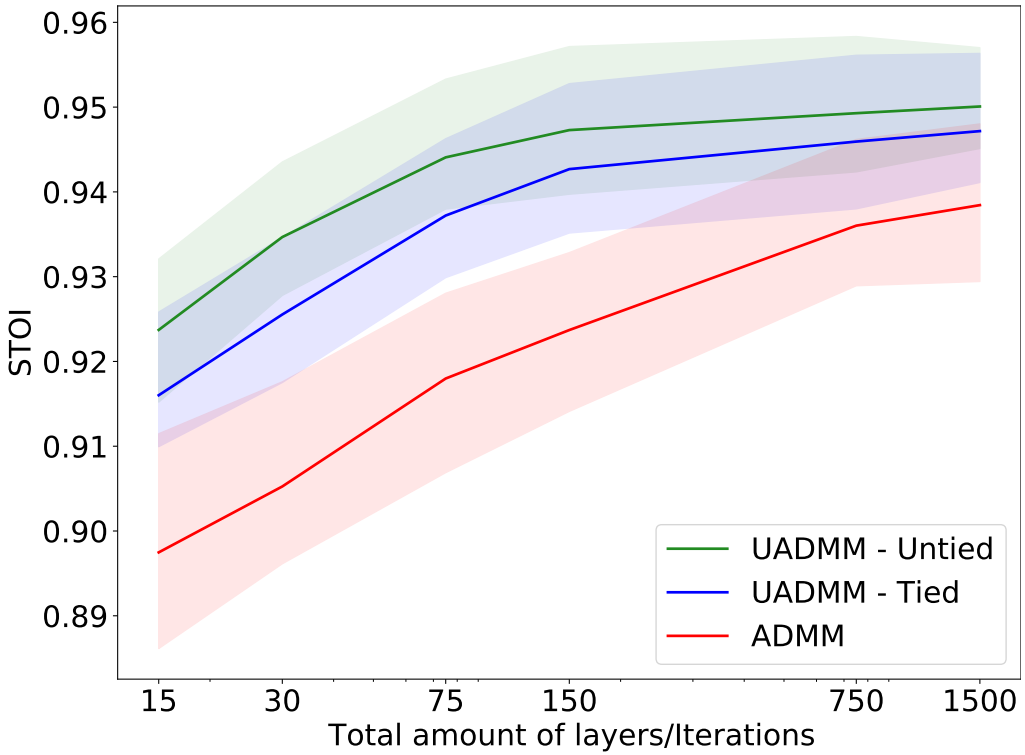


Training loss (negative STOI) over epochs



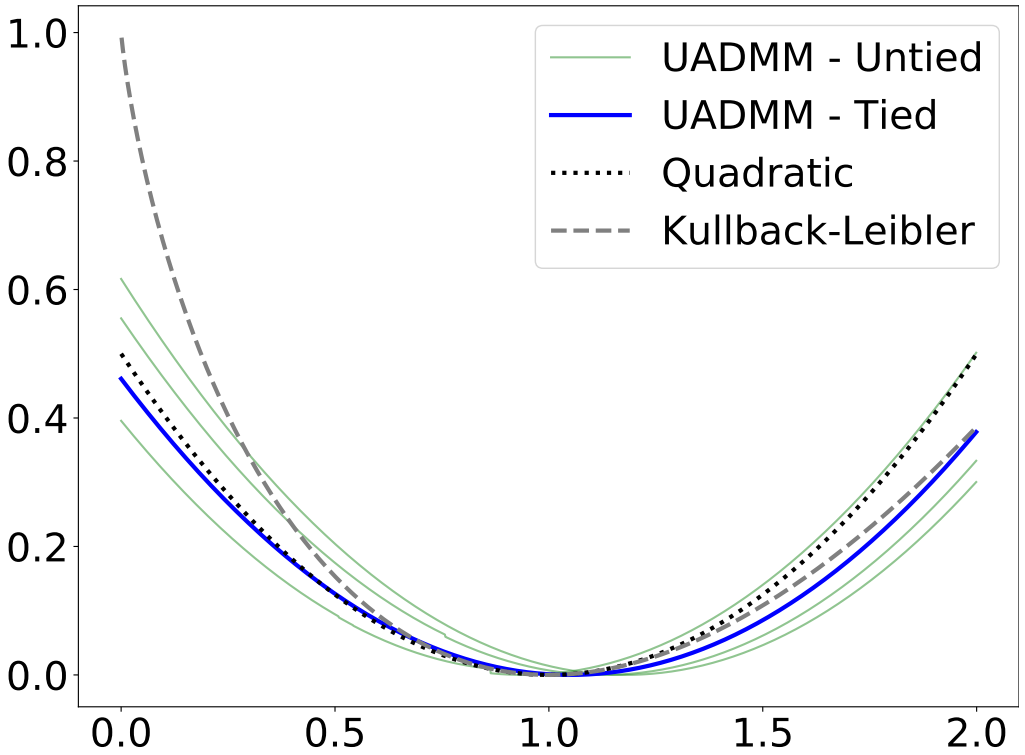
Performance on the test set

Iterating the network



Evaluation with STOI over test dataset with iterated model

Learned divergences



Learned metrics $f_{r,t}(y)$ with $r = 1$. The quadratic loss and Kullback-Leibler divergence $\mathcal{D}_{KL}(y \,|\, r)$ are also displayed for the sake of comparison. In the "tied" case, f_r in equal to $\mathcal{D}_{\psi}(\cdot \,|\, r)$ involved in the PR optimization problem. For clarity, only 3 of the 15 trained layers $f_{r,t}$ are displayed for the "untied" case.

Conclusion

Summary

- Unfolding allows to easily learn a divergence
- It can improve the results compared to a fixed loss such as ℓ_2
- Illustration for phase retrieval with unfolded ADMM

Perspectives

- Convergence in case *untied* ?
- Non-separable divergence
- Other inverse problems and settings

Thank you for listening

Any question ?

References

- Vial, Magron, Oberlin and Févotte, *Phase retrieval with Bregman divergences and application to audio signal recovery*, IEEE JSTSP, 2021
- Vial, Magron, Oberlin and Févotte, *Learning the Proximity Operator in Unfolded ADMM for Phase Retrieval*, IEEE SPL, 2022
- `https://github.com/phvial/LearningProxPR`