Learning Discrete Measurement Matrices and Algorithm Unfolding for Signal Recovery

Siyakhula - German Research Days at AIMS' 20-years Celebration, 2024

Peter Jung, Technical University Berlin
peter.jung@tu-berlin.de

March 22, 2024

Inverse problems: given observation $\boldsymbol{y}$ and partially **known** forward operator $\boldsymbol{A}$:

$$\text{find} \quad \boldsymbol{x} \in \mathcal{X} \quad \text{s.t.} \quad \boldsymbol{A}(\boldsymbol{x}) \approx \boldsymbol{y}$$

enforce structure for $\boldsymbol{x}$ via priors/regularization/models...

☺ Sparsity, low-rankness, atomic sets - **well-established theory**

Inverse problems: given observation $\boldsymbol{y}$ and partially **known** forward operator $\boldsymbol{A}$:

$$\text{find} \quad \boldsymbol{x} \in \mathcal{X} \quad \text{s.t.} \quad \boldsymbol{A}(\boldsymbol{x}) \approx \boldsymbol{y}$$

enforce structure for $\boldsymbol{x}$ via priors/regularization/models...

- ☺ Sparsity, low-rankness, atomic sets - **well-established theory**
- ☹ often **too generic** for real-world data
- ☹ algorithms $\boldsymbol{Q}$ are too **slow** and need handcrafted **tuning** for many real-world/time applications

Inverse problems: given observation $\boldsymbol{y}$ and partially **known** forward operator $\boldsymbol{A}$:

$$\text{find} \quad \boldsymbol{x} \in \mathcal{X} \quad \text{s.t.} \quad \boldsymbol{A}(\boldsymbol{x}) \approx \boldsymbol{y}$$

enforce structure for $\boldsymbol{x}$ via priors/regularization/models...

- ☺ Sparsity, low-rankness, atomic sets - **well-established theory**
- ☹ often **too generic** for real-world data
- ☹ algorithms $\boldsymbol{Q}$ are too **slow** and need handcrafted **tuning** for many real-world/time applications
- ☹ optimize $\boldsymbol{A}$ under **practical constraints** is often difficult

Inverse problems: given observation $\boldsymbol{y}$ and partially **known** forward operator $\boldsymbol{A}$:

$$\text{find} \quad \boldsymbol{x} \in \mathcal{X} \quad \text{s.t.} \quad \boldsymbol{A}(\boldsymbol{x}) \approx \boldsymbol{y}$$

enforce structure for $\boldsymbol{x}$ via priors/regularization/models...

- ☺ Sparsity, low-rankness, atomic sets - **well-established theory**
- ☹ often **too generic** for real-world data
- ☹ algorithms $\boldsymbol{Q}$ are too **slow** and need handcrafted **tuning** for many real-world/time applications
- ☹ optimize $\boldsymbol{A}$ under **practical constraints** is often difficult

"Learn to sense and to recover"

$$\text{find} \quad (\boldsymbol{A}, \boldsymbol{Q}) \qquad \text{s.t.} \quad \mathbb{E}_{\boldsymbol{x} \sim \mathcal{X}} \, \textbf{loss}(\boldsymbol{x}, \boldsymbol{Q}(\boldsymbol{A}(\boldsymbol{x}))) \quad \text{"is small"}$$

Outline

Gradient-Based Learning of Discrete Structured Measurement Operators for Signal Recovery

Jonathan Sauder, Martin Genzel, and Peter Jung

***Abstract*—Countless signal processing applications include the reconstruction of signals from few indirect linear measurements. The design of effective measurement operators is typically constrained by the underlying hardware and physics, posing a challenging and often even discrete optimization task. While the potential of gradient-based learning via the unrolling of iterative recovery algorithms has been demonstrated, it has remained unclear how to leverage this technique when the set of admissible measurement operators is structured and discrete. We tackle this problem by combining unrolled optimization with Gumbel reparametrizations, which enable the computation of**

discrete subset—to improve the performance of downstream tasks poses great computational challenges. While it is often easy to create a suitable random mask, it is not obvious how to optimize the measurement matrix in a way that is both efficient and respects the feasibility constraints. Classical approaches commonly use discrete optimization to find such sets, as no gradients can be directly computed.

On the other hand, gradient-based optimization via back-propagation through massive nonlinear computational graphs

Problem Statement

- Recover $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^n$ from (noisy) linear measurements $\boldsymbol{y} = \boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{\Phi}\boldsymbol{x} + \boldsymbol{e} \in \mathbb{R}^m$

- Recover $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^n$ from (noisy) linear measurements $\boldsymbol{y} = \boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{\Phi}\boldsymbol{x} + \boldsymbol{e} \in \mathbb{R}^m$
- minimal acquisition time&storage $m \simeq \mathrm{DoF}(\mathcal{X}) \ll n \Rightarrow$ **“sensing & compression”**

- Recover $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^n$ from (noisy) linear measurements $\boldsymbol{y} = \boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{\Phi}\boldsymbol{x} + \boldsymbol{e} \in \mathbb{R}^m$
- minimal acquisition time&storage $m \simeq \mathrm{DoF}(\mathcal{X}) \ll n \Rightarrow$ **"sensing & compression"**
- $\boldsymbol{\Phi}$ lies in **finite set of admissible matrices** $\boldsymbol{\Phi}$ (determined by physics / detectors)
- $\boldsymbol{x}$ is recovered from $\boldsymbol{y}$ using parametrized recovery algorithm $f_{\boldsymbol{\theta},\boldsymbol{\Phi}} : \mathbb{R}^m \to \mathbb{R}^n$
- $\boldsymbol{x} \to f_{\boldsymbol{\theta},\boldsymbol{\Phi}}(\boldsymbol{\Phi}\boldsymbol{x})$ can be understood as autoencoder

- Recover $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^n$ from (noisy) linear measurements $\boldsymbol{y} = \boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{\Phi}\boldsymbol{x} + \boldsymbol{e} \in \mathbb{R}^m$
- minimal acquisition time&storage $m \simeq \mathrm{DoF}(\mathcal{X}) \ll n \Rightarrow$ **"sensing & compression"**
- $\boldsymbol{\Phi}$ lies in **finite set of admissible matrices** $\boldsymbol{\mathit{\Phi}}$ (determined by physics / detectors)
- $\boldsymbol{x}$ is recovered from $\boldsymbol{y}$ using parametrized recovery algorithm $f_{\boldsymbol{\theta},\boldsymbol{\Phi}} : \mathbb{R}^m \to \mathbb{R}^n$
- $\boldsymbol{x} \to f_{\boldsymbol{\theta},\boldsymbol{\Phi}}(\boldsymbol{\Phi}\boldsymbol{x})$ can be understood as autoencoder
- find good and **admissible** $\boldsymbol{\Phi} \in \boldsymbol{\mathit{\Phi}}$ and algorithm parameters $\boldsymbol{\theta}$

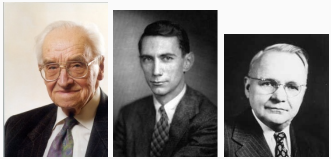
$$\min_{\boldsymbol{\Phi}\in\boldsymbol{\mathit{\Phi}},\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{X}} \left[\mathbf{loss}(f_{\boldsymbol{\theta},\boldsymbol{\Phi}}(\boldsymbol{\Phi}\boldsymbol{x}), \boldsymbol{x}) \right]$$

data-driven via "training"

prototypical playground is **"compressed sensing"** $\Leftrightarrow$ recovering sparse $\boldsymbol{x}$ from $\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x} + \boldsymbol{e}$

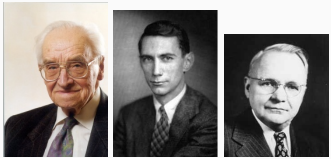
classical sampling/scanning *"…if you sample densely enough, you can perfectly reconstruct the original analog data…"*

- ☺ good, if signals "fill up" a subspace (linear structure)
- ☹ wasteful for compressible signals (non–linear structure)

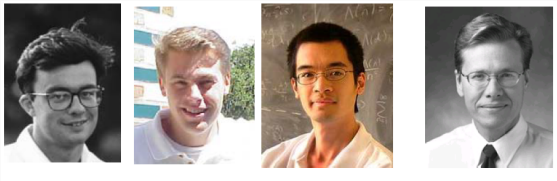


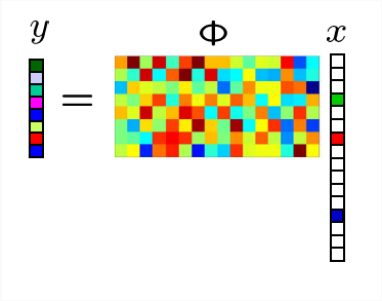
pictogram from R. Baraniuk, "Compressive Sensing", Eusipco09

classical sampling/scanning *"…if you sample densely enough, you can perfectly reconstruct the original analog data…"*



- ☺ good, if signals "fill up" a subspace (linear structure)
- ☹ wasteful for compressible signals (non–linear structure)

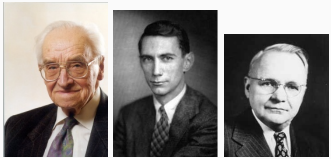




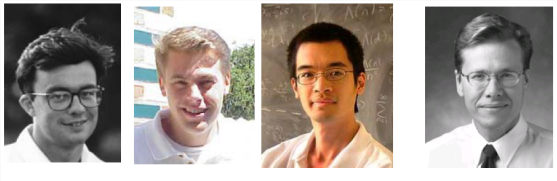
"…Can we not just directly measure the part that will not end up being thrown away ?" [Donoho, 2006]

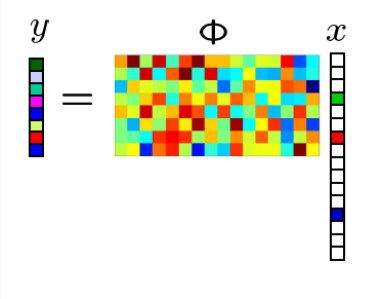
pictogram from R. Baraniuk, "Compressive Sensing", Eusipco09

classical sampling/scanning *"…if you sample densely enough, you can perfectly reconstruct the original analog data…"*



- ☺ good, if signals "fill up" a subspace (linear structure)
- ☹ wasteful for compressible signals (non–linear structure)





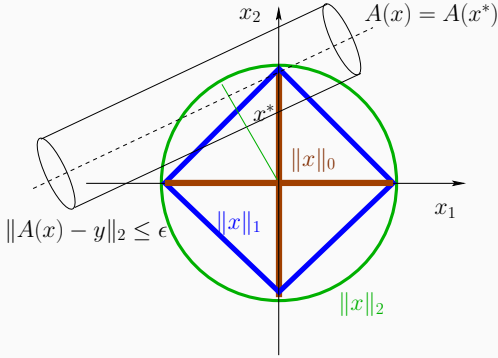
"…Can we not just directly measure the part that will not end up being thrown away ?" [Donoho, 2006]

- ☺ Well-investigated, theoretical bounds, algorithms

pictogram from R. Baraniuk, "Compressive Sensing", Eusipco09

Convex recovery approach with guarantees

$$x^\sharp = \arg\min \|z\|_1 \quad \text{s.t.} \quad \|\Phi z - y\| \leq \epsilon$$



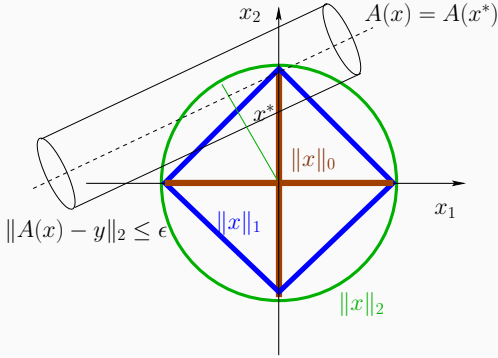
Convex recovery approach with guarantees

$$x^\sharp = \arg\min \|z\|_1 \quad \text{s.t.} \quad \|\Phi z - y\| \leq \epsilon$$

if sparse vectors are "well-separated" from nullspace of Φ (NSP $\Leftarrow$ RIP $\Leftarrow$ coherence):

$$\left\|x^\sharp - x\right\| \lesssim \epsilon$$

holds for all $y = \Phi x + e$ with $\|e\| \leq \epsilon$.



- depends on hyper-parameter ϵ (in this case the noise-level/SNR)
- ☹ Problematic, if noise depends on x (Poisson/multiplicative noise, covariance matching…)

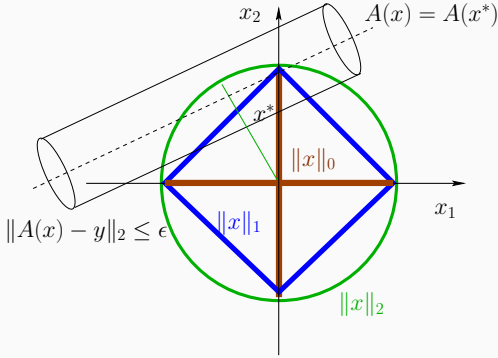
Convex recovery approach with guarantees

$$x^\sharp = \arg\min \|z\|_1 \quad \text{s.t.} \quad \|\Phi z - y\| \leq \epsilon$$

if sparse vectors are "well-separated" from nullspace of Φ (NSP $\Leftarrow$ RIP $\Leftarrow$ coherence):

$$\left\|x^\sharp - x\right\| \lesssim \epsilon$$

holds for all $y = \Phi x + e$ with $\|e\| \leq \epsilon$.



- depends on hyper-parameter ϵ (in this case the noise-level/SNR)
- ☹ Problematic, if noise depends on x (Poisson/multiplicative noise, covariance matching…)
- ☺ Non-negativity & "biased" measurements are helpful, example later

task 1: find $\Phi \in \mathbb{R}^{m \times n}$ with minimal m for given sparsity s under practical constraints

task 1: find $\mathbf{\Phi} \in \mathbb{R}^{m \times n}$ with minimal m for given sparsity s under practical constraints

- deterministic constructions known for $m = O(s^2)$ (based on coherence)
- "sufficiently random" whp for $m = O(s \operatorname{polylog}(n))$ (RIP/NSP etc.) [Candes & Tao, 2005]
- …many works on other random models …

task 1: find $\mathbf{\Phi} \in \mathbb{R}^{m \times n}$ with minimal m for given sparsity s under practical constraints

- deterministic constructions known for $m = O(s^2)$ (based on coherence)
- "sufficiently random" whp for $m = O(s \operatorname{polylog}(n))$ (RIP/NSP etc.) [Candes & Tao, 2005]
- ...many works on other random models ...
- random binary matrices [Kueng & PJ, 2018], designs and orthogonal arrays [PJ, Kueng, Mixton , 2019]

$\Rightarrow$ "structured derandomization", find "concentrated measure" p on $\boldsymbol{\Phi}$

task 1: find $\mathbf{\Phi} \in \mathbb{R}^{m \times n}$ with minimal m for given sparsity s under practical constraints

- deterministic constructions known for $m = O(s^2)$ (based on coherence)
- "sufficiently random" whp for $m = O(s \operatorname{polylog}(n))$ (RIP/NSP etc.) [Candes & Tao, 2005]
- ...many works on other random models ...
- random binary matrices [Kueng & PJ, 2018], designs and orthogonal arrays [PJ, Kueng, Mixton , 2019]

$\Rightarrow$ "structured derandomization", find "concentrated measure" p on $\boldsymbol{\Phi}$

task 2: develop fast algorithms and hyper-parameter tuning

- most of algorithms are too complex and not suited for real-time applications

task 1: find $\mathbf{\Phi} \in \mathbb{R}^{m \times n}$ with minimal m for given sparsity s under practical constraints

- deterministic constructions known for $m = O(s^2)$ (based on coherence)
- "sufficiently random" whp for $m = O(s \operatorname{polylog}(n))$ (RIP/NSP etc.) [Candes & Tao, 2005]
- ...many works on other random models ...
- random binary matrices [Kueng & PJ, 2018], designs and orthogonal arrays [PJ, Kueng, Mixton , 2019]

$\Rightarrow$ "structured derandomization", find "concentrated measure" p on $\boldsymbol{\Phi}$

task 2: develop fast algorithms and hyper-parameter tuning

- most of algorithms are too complex and not suited for real-time applications

in "machine learning" terminology:

$$\min_{p,\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x},\mathbf{\Phi}\sim p}[\mathbf{loss}(f_{\boldsymbol{\theta},\mathbf{\Phi}}(\mathbf{\Phi}x), x)]$$

two techniques: **Gumbel reparametrizations** and **algorithm unfolding**

Gradient-based Learning of Discrete Measurement Matrices

- learning **unstructured** $\Phi \Rightarrow$ use ∇_{Φ} [Adler et al 2016, Wu et al. 2019]

- learning **unstructured** $\boldsymbol{\Phi} \Rightarrow$ use $\nabla_{\boldsymbol{\Phi}}$ [Adler et al 2016, Wu et al. 2019]
- here $\boldsymbol{\Phi}$ is **discrete**, i.e., optimize instead **a discrete distribution** p **on** $\boldsymbol{\Phi}$

$$\min_{p,\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x},\boldsymbol{\Phi}\sim p}[\mathbf{loss}(f_{\boldsymbol{\theta},\boldsymbol{\Phi}}(\boldsymbol{\Phi}\boldsymbol{x}), \boldsymbol{x})]$$

aligns with derandomization principle

- learning **unstructured** $\mathbf{\Phi} \Rightarrow$ use $\nabla_{\mathbf{\Phi}}$ [Adler et al 2016, Wu et al. 2019]
- here $\mathbf{\Phi}$ is **discrete**, i.e., optimize instead **a discrete distribution p on $\mathbf{\Phi}$**

$$\min_{p,\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x},\mathbf{\Phi}\sim p}[\mathbf{loss}(f_{\boldsymbol{\theta},\mathbf{\Phi}}(\mathbf{\Phi}\boldsymbol{x}), \boldsymbol{x})]$$

aligns with derandomization principle

1. Gradient descent on p but estimate $\nabla_p \mathbb{E}_{\mathbf{\Phi}\sim p} f(\mathbf{\Phi})$ via **sampling**
2. perform sampling in a way that allows backprop - **Gumbel reparametrization**

- Consider computational graph with a *random node* v taking the values in $[a] := \{1, \dots, a\}$.
- inputs to node v are unnormalized log-probabilities $\boldsymbol{\varphi} = (\varphi_1 \ \dots, \varphi_a) \in \mathbb{R}^a$

$$\mathbb{P}(v = i) = \frac{\exp(\varphi_i)}{\sum_{j=1}^{a} \exp(\varphi_j)} = \mathsf{softmax}(\boldsymbol{\varphi})_i, \quad i = 1, \dots, a$$

- realization of v is then passed through a differentiable function f.

Estimate $\nabla_\varphi \mathbb{E}_v \Big[f(v) \Big]$ by sampling

- Consider computational graph with a *random node* v taking the values in $[a] := \{1, \dots, a\}$.
- inputs to node v are unnormalized log-probabilities $\boldsymbol{\varphi} = (\varphi_1 \dots, \varphi_a) \in \mathbb{R}^a$

$$\mathbb{P}(v = i) = \frac{\exp(\varphi_i)}{\sum_{j=1}^{a} \exp(\varphi_j)} = \mathsf{softmax}(\boldsymbol{\varphi})_i, \quad i = 1, \dots, a$$

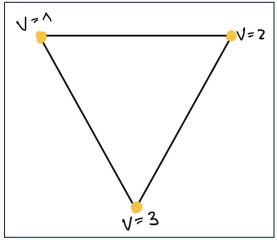
- realization of v is then passed through a differentiable function f.

Estimate $\nabla_\varphi \mathbb{E}_v \Big[f(v) \Big]$ by sampling

- efficiently sample from v by using [Gumbel 1954]

$$\boldsymbol{v} = \mathsf{one\_hot}(\arg\max_i [\varphi_i + g_i])$$

where $g_i = -\log(-\log(u_i))$ with $u_1, \dots, u_a \sim_{\text{iid}} \mathsf{unif}([0, 1])$



…so, we can sample v via

$$v = \text{one\_hot}(\arg\max_i [\varphi_i + g_i]) \qquad g_i \sim G(0, 1)$$

☹ But backprop doesn't work because of $\arg\max$.

…so, we can sample v via

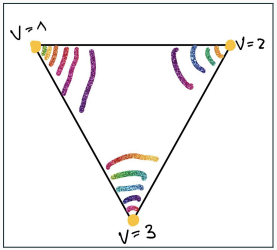
$$v = \text{one\_hot}(\arg\max_i [\varphi_i + g_i]) \qquad g_i \sim G(0,1)$$

☹ But backprop doesn't work because of $\arg\max$.

☺ **Gumbel-softmax** [Jang et al. 2016; Maddison et al. 2016]
replace $\arg\max$ in backward pass with softmax with temperature τ:

$$v_i = \frac{\exp((g_i + \varphi_i)/\tau)}{\sum_{j=1}^{a} \exp((g_j + \varphi_j)/\tau)} = \text{softmax}((\boldsymbol{\varphi} + \boldsymbol{g})/\tau)_i$$

- As $\tau \to 0$, softmax approaches $\arg\max$



- from reservoir sampling: Gumbel softmax extends naturally to Gumbel top-k! [Vieira 2014]

☺ learnable component for "selecting k out of a set"

The Method (for structured binary $m \times n$-matrix)

- from reservoir sampling: Gumbel softmax extends naturally to Gumbel top-k! [Vieira 2014]
- ☺ learnable component for "selecting k out of a set"

The Method (for structured binary $m \times n$-matrix)

1. let $\{I_i\}_{i \in [l]}$ a partition of $\mathcal{I} := [m] \times [n]$

- from reservoir sampling: Gumbel softmax extends naturally to Gumbel top-k! [Vieira 2014]
- ☺ learnable component for "selecting k out of a set"

The Method (for structured binary $m \times n$-matrix)

1. let $\{I_i\}_{i \in [l]}$ a partition of $\mathcal{I} := [m] \times [n]$
2. for each $i \in [l]$ use Gumbel top-k to select k_i elements from I_i.

- from reservoir sampling: Gumbel softmax extends naturally to Gumbel top-k! [Vieira 2014]

☺ learnable component for "selecting k out of a set"

The Method (for structured binary $m \times n$-matrix)

1. let $\{I_i\}_{i \in [l]}$ a partition of $\mathcal{I} := [m] \times [n]$
2. for each $i \in [l]$ use Gumbel top-k to select k_i elements from I_i.
3. forward pass: gives a random $m \times n$ binary matrix $\mathbf{\Phi}$ via top-k_i and ...

- from reservoir sampling: Gumbel softmax extends naturally to Gumbel top-k! [Vieira 2014]

☺ learnable component for "selecting k out of a set"

The Method (for structured binary $m \times n$-matrix)

1. let $\{I_i\}_{i \in [l]}$ a partition of $\mathcal{I} := [m] \times [n]$
2. for each $i \in [l]$ use Gumbel top-k to select k_i elements from I_i.
3. forward pass: gives a random $m \times n$ binary matrix $\mathbf{\Phi}$ via top-k_i and ...
4. backward pass: well-defined gradient prop. via softmax instead of hard top-k_i [Bengio et.al, 2013]
5. construction can be freely used in conjunction with automatic differentiation

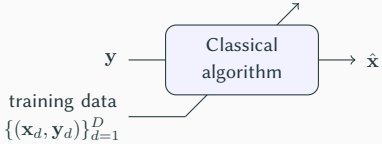
- from reservoir sampling: Gumbel softmax extends naturally to Gumbel top-k! [Vieira 2014]
- ☺ learnable component for "selecting k out of a set"

The Method (for structured binary $m \times n$-matrix)

1. let $\{I_i\}_{i \in [l]}$ a partition of $\mathcal{I} := [m] \times [n]$
2. for each $i \in [l]$ use Gumbel top-k to select k_i elements from I_i.
3. forward pass: gives a random $m \times n$ binary matrix $\mathbf{\Phi}$ via top-k_i and ...
4. backward pass: well-defined gradient prop. via softmax instead of hard top-k_i [Bengio et.al, 2013]
5. construction can be freely used in conjunction with automatic differentiation

ok, now we also need a "trainable" recovery algorithm ...

Algorithm Unfolding



Computational graph of many iterative algorithms can be viewed as neural networks…

- consider for example:

$$\min_{x} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2 + \lambda g(\boldsymbol{x})$$

and use proximal (Landweber) iterations:

$$\hat{\boldsymbol{x}}^{(t+1)} = \text{prox}_{\lambda g}\Big(\hat{\boldsymbol{x}}^{(t)} + \gamma \nabla\big(\|\boldsymbol{y} - \boldsymbol{\Phi}\hat{\boldsymbol{x}}^{(t)}\|_2^2\big)\Big),$$

Computational graph of many iterative algorithms can be viewed as neural networks…

- consider for example:

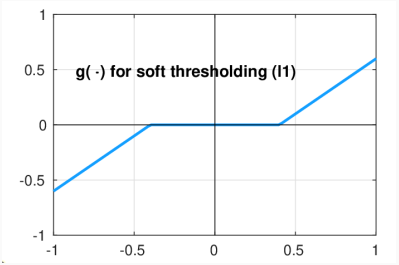
$$\min_x \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2 + \lambda g(\boldsymbol{x})$$

and use proximal (Landweber) iterations:

$$\hat{\boldsymbol{x}}^{(t+1)} = \text{prox}_{\lambda g}\Big(\hat{\boldsymbol{x}}^{(t)} + \gamma\nabla\big(\|\boldsymbol{y} - \boldsymbol{\Phi}\hat{\boldsymbol{x}}^{(t)}\|_2^2\big)\Big),$$

- $g(\boldsymbol{x}) = \|\boldsymbol{x}\|_1 \Rightarrow$ element-wise soft-thresholding, known as *Iterative Shrinkage and Thresholding Algorithm* [Daubechies et al., 2004]

(ISTA) $$\hat{\boldsymbol{x}}^{(t+1)} = \eta_{\frac{\lambda}{L}}\Big(\hat{\boldsymbol{x}}^{(t)} + \gamma\boldsymbol{\Phi}^*(\boldsymbol{\Phi}\hat{\boldsymbol{x}}^{(t)} - \boldsymbol{y})\Big)$$



Unfolding iterations into network $f_{\boldsymbol{\theta},\boldsymbol{\Phi}}$, several proposals [Gregor&LeCun 2010, Liu 2019, Chen 2018,...]:

(LISTA-CP) $$\boldsymbol{x}^{(k+1)} = \eta_{\alpha^{(k)}}\left(\boldsymbol{x}^{(k)} - \boldsymbol{B}^{(k)^*}\left(\boldsymbol{\Phi}\boldsymbol{x}^{(k)} - \boldsymbol{y}\right)\right) \qquad \{\alpha^{(k)}, \boldsymbol{B}^{(k)}\}$$

Unfolding iterations into network $f_{\boldsymbol{\theta},\boldsymbol{\Phi}}$, several proposals [Gregor&LeCun 2010, Liu 2019, Chen 2018,...]:

(LISTA-CP) $$\boldsymbol{x}^{(k+1)} = \eta_{\alpha^{(k)}}\left(\boldsymbol{x}^{(k)} - \boldsymbol{B}^{(k)^*}\left(\boldsymbol{\Phi}\boldsymbol{x}^{(k)} - \boldsymbol{y}\right)\right) \qquad \{\alpha^{(k)}, \boldsymbol{B}^{(k)}\}$$

(ALISTA) $$\boldsymbol{x}^{(k+1)} = \eta_{\alpha^{(k)}}\left(\boldsymbol{x}^{(k)} - \gamma^{(k)}\boldsymbol{B}^*\left(\boldsymbol{\Phi}\boldsymbol{x}^{(k)} - \boldsymbol{y}\right)\right) \qquad \{\alpha^{(k)}, \gamma^{(k)}\}$$

- supervised training on $\mathcal{X}$ with $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{X},\boldsymbol{e}}\,\mathbf{loss}(f_{\boldsymbol{\theta},\boldsymbol{\Phi}}(\boldsymbol{\Phi}\boldsymbol{x}+\boldsymbol{e}),\boldsymbol{x})$, unsupervised is possible as well

Unfolding iterations into network $f_{\boldsymbol{\theta},\boldsymbol{\Phi}}$, several proposals [Gregor&LeCun 2010, Liu 2019, Chen 2018,...]:

(LISTA-CP) $$\boldsymbol{x}^{(k+1)} = \eta_{\alpha^{(k)}}\left(\boldsymbol{x}^{(k)} - \boldsymbol{B}^{(k)^*}\left(\boldsymbol{\Phi}\boldsymbol{x}^{(k)} - \boldsymbol{y}\right)\right) \qquad \{\alpha^{(k)}, \boldsymbol{B}^{(k)}\}$$

(ALISTA) $$\boldsymbol{x}^{(k+1)} = \eta_{\alpha^{(k)}}\left(\boldsymbol{x}^{(k)} - \gamma^{(k)}\boldsymbol{B}^*\left(\boldsymbol{\Phi}\boldsymbol{x}^{(k)} - \boldsymbol{y}\right)\right) \qquad \{\alpha^{(k)}, \gamma^{(k)}\}$$

- supervised training on $\mathcal{X}$ with $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{X},\boldsymbol{e}}$ **loss**$(f_{\boldsymbol{\theta},\boldsymbol{\Phi}}(\boldsymbol{\Phi}\boldsymbol{x}+\boldsymbol{e}),\boldsymbol{x})$, unsupervised is possible as well
- $\mathcal{X}$ = bounded s-sparse vectors and cross-coherence $\mu := \mu(\boldsymbol{B},\boldsymbol{\Phi})$

Theorem: [Liu et al., 2019 ; Hauffen, PJ, Mücke 2022]

For any $\boldsymbol{x} \in \mathcal{X}$ with $s < (\mu^{-1}+1)/2$, $\|\boldsymbol{e}\|_2 \leq \epsilon$ and $\alpha^{(k)} \gtrsim \mu + \epsilon$ **ALISTA** yields:

$$\|\boldsymbol{x}^{(k)} - \boldsymbol{x}\|_2 \lesssim c^k s + \left(1 + kc^k\right)\|\boldsymbol{e}\|_2$$

for $c = c(\{\alpha_{k'}\}_{k'=1}^k) < 1$ and constants depend on $\mathcal{X}$ and $\{\alpha_{k'}\}_{k'=1}^k$.

- adaptive parameters depend on $\|\boldsymbol{x}^{(k)} - \boldsymbol{x}\|_1$

- adaptive parameters depend on $\|\boldsymbol{x}^{(k)} - \boldsymbol{x}\|_1$
- use $r^{(k)} = \|\boldsymbol{\Phi}\boldsymbol{x}^{(k)} - y\|_1$ and $u^{(k)} = \|\boldsymbol{B}^*(\boldsymbol{\Phi}\boldsymbol{x}^{(k)} - y)\|_1$
- *neurally augmented ALISTA* (NA-ALISTA) [Behrens, Sauder, PJ 2020]:

$$\boldsymbol{x}^{(k+1)} = \eta_{\alpha(r^{(k)}, u^{(k)})}\left(\boldsymbol{x}^{(k)} - \gamma(r^{(k)}, u^{(k)})\left[\boldsymbol{B}^*\left(\boldsymbol{\Phi}\boldsymbol{x}^{(k)} - \boldsymbol{y}\right)\right]\right)$$

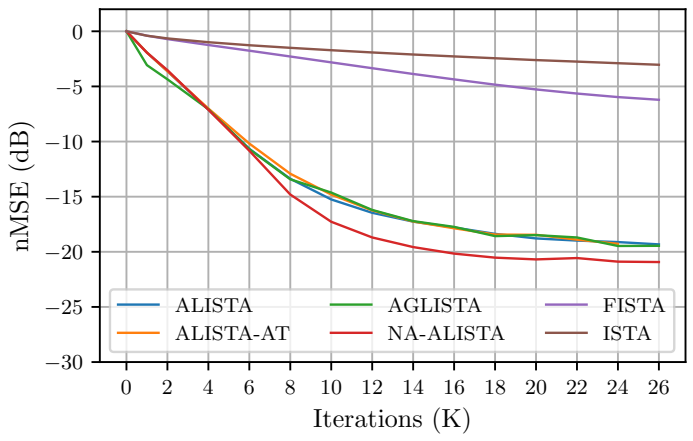
train LSTMs to represent $\alpha(\cdot, \cdot)$ and $\gamma(\cdot, \cdot)$

- adaptive parameters depend on $\|\boldsymbol{x}^{(k)} - \boldsymbol{x}\|_1$
- use $r^{(k)} = \|\boldsymbol{\Phi}\boldsymbol{x}^{(k)} - y\|_1$ and $u^{(k)} = \|\boldsymbol{B}^*(\boldsymbol{\Phi}\boldsymbol{x}^{(k)} - y)\|_1$
- *neurally augmented ALISTA* (NA-ALISTA) [Behrens, Sauder, PJ 2020]:

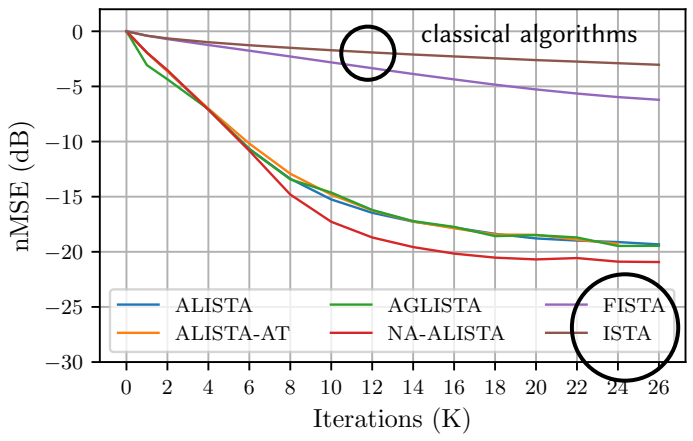
$$\boldsymbol{x}^{(k+1)} = \eta_{\alpha(r^{(k)}, u^{(k)})}\left(\boldsymbol{x}^{(k)} - \gamma(r^{(k)}, u^{(k)})\left[\boldsymbol{B}^*\left(\boldsymbol{\Phi}\boldsymbol{x}^{(k)} - \boldsymbol{y}\right)\right]\right)$$

train LSTMs to represent $\alpha(\cdot, \cdot)$ and $\gamma(\cdot, \cdot)$

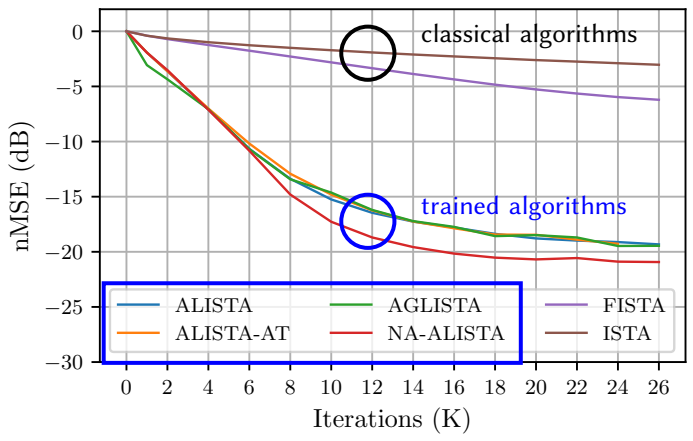
Significantly reduces number of iterations required! Enabling real-time applications



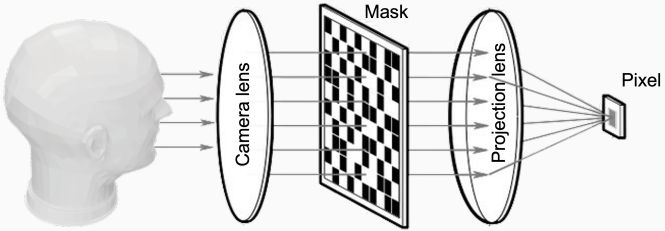
Significantly reduces number of iterations required! Enabling real-time applications



Significantly reduces number of iterations required! Enabling real-time applications



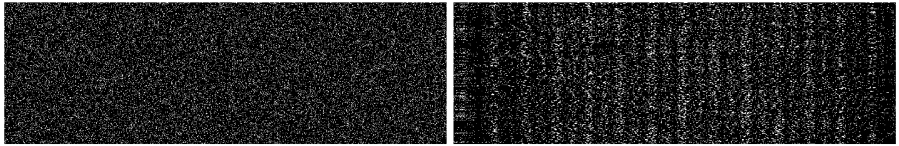
Application 1: Learning Masks for Single-Pixel Imaging



[adapted from [Bacca et al. 2019]]

- $\mathcal{X}$ =MNIST ($n = 28^2 = 784$)
- $\boldsymbol{\Phi}$ is the set of **binary matrices** with $k_i = 32$ ones per row i ("on"-pixels per mask)
- $f_{\boldsymbol{\theta},\boldsymbol{\Phi}}$ is NA-ALISTA with $T = 20$ layers/iterations
- $\mathcal{L}(\hat{x}, x) = \|\hat{x} - x\|_2^2$

(a) Random Masks (b) Learned Masks

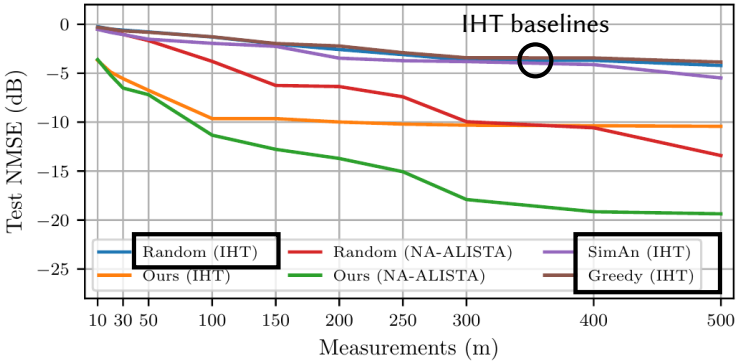


(c) Random Φ (d) Learned Φ

(e) Reconstructed (random Φ) (f) Reconstructed (learned Φ)

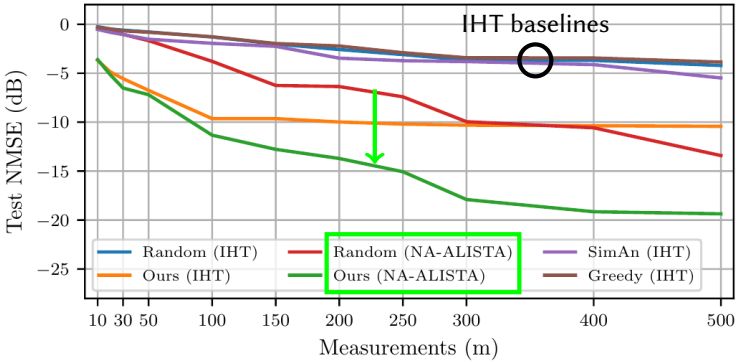
baseline: Iterative Hard-thresholding (IHT) $\boldsymbol{x}^{(k+1)} = H_s\left(\boldsymbol{x}^{(k)} - \boldsymbol{\Phi}^*\left(\boldsymbol{\Phi}\boldsymbol{x}^{(k)} - \boldsymbol{y}\right)\right)$

- $n = 28^2 = 784$ ($\mathcal{X}$=MNIST)
- random $I \subset \mathcal{I} = [m] \times [n]$
- swap a random $1 \Leftrightarrow 0$
- accept if loss L decreases
- **Greedy:** not accept if L increase
- **SimAn:** accept if $\exp[(L - L')/\tau] \leq u \sim U(0,1)$ and $\tau \to 0$ with training



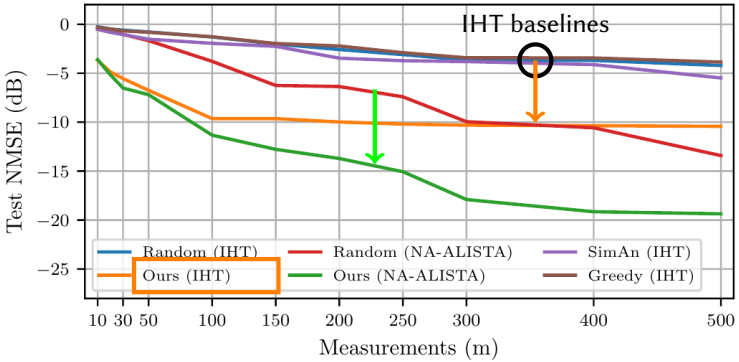
baseline: Iterative Hard-thresholding (IHT) $\boldsymbol{x}^{(k+1)} = H_s\left(\boldsymbol{x}^{(k)} - \boldsymbol{\Phi}^*\left(\boldsymbol{\Phi}\boldsymbol{x}^{(k)} - \boldsymbol{y}\right)\right)$

- $n = 28^2 = 784$ ($\mathcal{X}$=MNIST)
- random $I \subset \mathcal{I} = [m] \times [n]$
- swap a random $1 \Leftrightarrow 0$
- accept if loss L decreases
- **Greedy:** not accept if L increase
- **SimAn:** accept if $\exp[(L - L')/\tau] \leq u \sim U(0,1)$ and $\tau \to 0$ with training

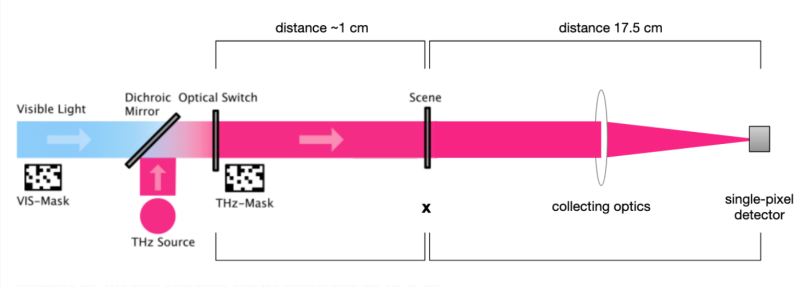


baseline: Iterative Hard-thresholding (IHT) $\boldsymbol{x}^{(k+1)} = H_s\left(\boldsymbol{x}^{(k)} - \boldsymbol{\Phi}^*\left(\boldsymbol{\Phi}\boldsymbol{x}^{(k)} - \boldsymbol{y}\right)\right)$

- $n = 28^2 = 784$ ($\mathcal{X}$=MNIST)
- random $I \subset \mathcal{I} = [m] \times [n]$
- swap a random $1 \Leftrightarrow 0$
- accept if loss L decreases
- **Greedy:** not accept if L increase
- **SimAn:** accept if $\exp[(L - L')/\tau] \leq u \sim U(0,1)$ and $\tau \to 0$ with training

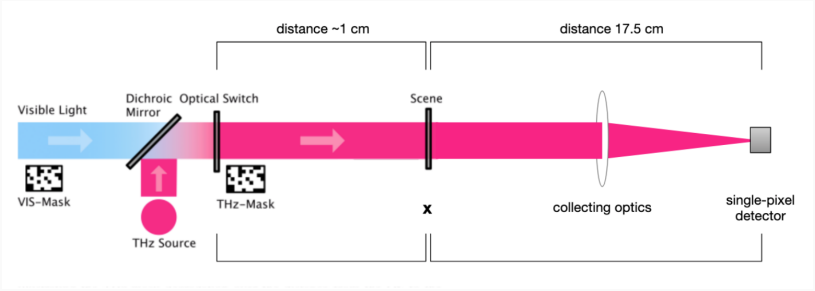


☺ **Improvements for IHT while trained with NA-ALISTA ⇒ due to Φ**

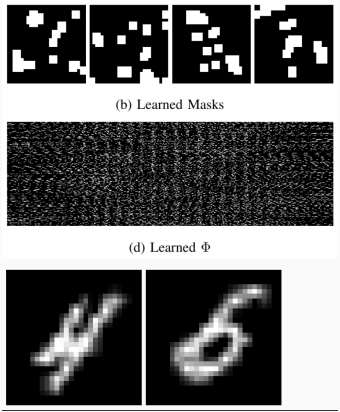


Terahertz SPI [Augustin, PJ, Frohmann, Hübers, 2019] - image: [Reiche, PJ 2020]

- wavelength is in order of pixel size, diffraction degrades image quality ⇒ use super-pixels!



Terahertz SPI [Augustin, PJ, Frohmann, Hübers, 2019] - image: [Reiche, PJ 2020]



- wavelength is in order of pixel size, diffraction degrades image quality ⇒ use super-pixels!
- sample binary masks with d **ones forming super-pixels of size** Δ

Application 2: Learning Pooling Matrices for Group Testing

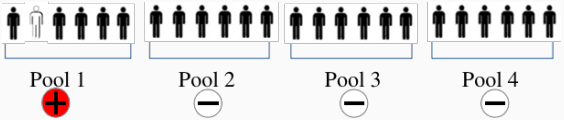
- **Non-adaptive viral testing of n individuals with minimal amount of m qPCR-tests**
- $s \ll n$ individuals are positive, i.e., vector of viral loads $\boldsymbol{x} = (x_1, \ldots, x_n)$ **is sparse**

  thus, viral-load recovery is a compressed sensing problem [Bah, Petersen & PJ, 2024]

- **Non-adaptive viral testing of n individuals with minimal amount of m qPCR-tests**
- $s \ll n$ individuals are positive, i.e., vector of viral loads $\boldsymbol{x} = (x_1, \ldots, x_n)$ **is sparse**

  thus, viral-load recovery is a compressed sensing problem [Bah, Petersen & PJ, 2024]

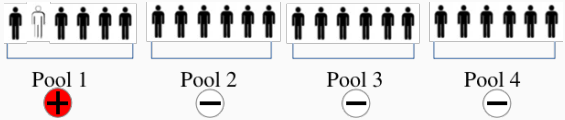
adaptive testing, $n = 24$, $m = 10$

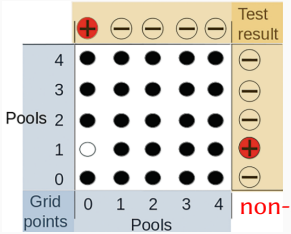


- **Non-adaptive viral testing of n individuals with minimal amount of m qPCR-tests**
- $s \ll n$ individuals are positive, i.e., vector of viral loads $\boldsymbol{x} = (x_1, \ldots, x_n)$ **is sparse**

  thus, viral-load recovery is a compressed sensing problem [Bah, Petersen & PJ, 2024]

adaptive testing, $n = 24, m = 10$





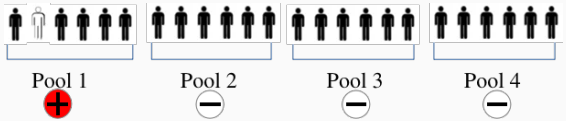
non-adaptive testing

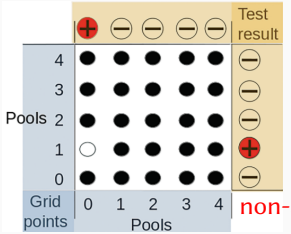
$n = 25, m = 10$

- **Non-adaptive viral testing of n individuals with minimal amount of m qPCR-tests**
- $s \ll n$ individuals are positive, i.e., vector of viral loads $\boldsymbol{x} = (x_1, \ldots, x_n)$ **is sparse**

  thus, viral-load recovery is a compressed sensing problem [Bah, Petersen & PJ, 2024]

adaptive testing, $n = 24, m = 10$





non-adaptive testing

$n = 25, m = 10$

- binary pooling matrix $\boldsymbol{\Phi} \in \{0,1\}^{m \times n}$ has k non-zeros per row (pool size)
- pooled measurements of viral loads are $\boldsymbol{y} = \boldsymbol{\Phi x} + \boldsymbol{e}$ with sparse $\boldsymbol{x} \geq 0$

- qPCR modeling is complicated and involves multiplicative noise, yielding heavy-tailed models
- **non-negative least absolute deviations** (NNLAD) is a compressed sensing algorithm

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x} \geq 0} \|\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{y}\|_1$$

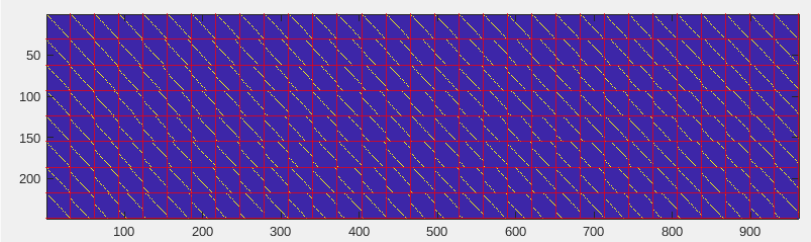
  if $\exists \boldsymbol{t}$ such that $\boldsymbol{\Phi}^T \boldsymbol{t} > 0$ and $\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_1 \lesssim O(\|\boldsymbol{e}\|_1)$ if $\boldsymbol{\Phi}$ has NSP [Petersen, Bah, PJ, 2021]
- ℓ_1-regularization is superflous $\Rightarrow$ **no hyperparameter tuning !**

- qPCR modeling is complicated and involves multiplicative noise, yielding heavy-tailed models
- **non-negative least absolute deviations** (NNLAD) is a compressed sensing algorithm

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x} \geq 0} \|\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{y}\|_1$$

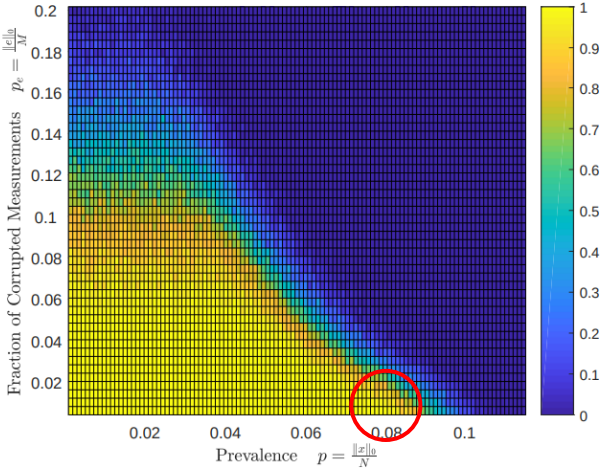
if $\exists \boldsymbol{t}$ such that $\boldsymbol{\Phi}^T \boldsymbol{t} > 0$ and $\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_1 \lesssim O(\|\boldsymbol{e}\|_1)$ if $\boldsymbol{\Phi}$ has NSP [Petersen, Bah, PJ, 2021]

- ℓ_1-regularization is superflous $\Rightarrow$ **no hyperparameter tuning !**



Example: disjunct matrix $\boldsymbol{\Phi} \in \{0,1\}^{248\times 961}$ with $k = 31$ and has NSP for $s \leq 7$ (LDPC/Array code) [Lofti&Vidyasagar, 2020]

robust against corrupted tests (sparse noise) and **empirical success up to** $s \lesssim 70$ **!!**



Learning pooling matrices (k ones per row) and tuning viral load recovery

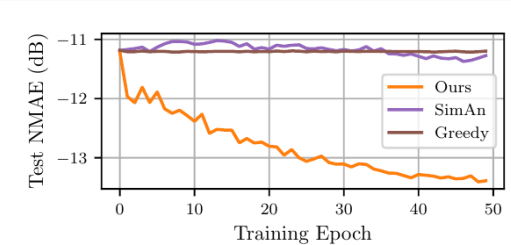
- projected subgradient descent for NNLAD can be unfolded
- slow convergence (not relevant for testing)
- supervised training with 200 iterations/layers
- inference with 1000 iterations

Learning pooling matrices (k ones per row) and tuning viral load recovery

- projected subgradient descent for NNLAD can be unfolded
- slow convergence (not relevant for testing)
- supervised training with 200 iterations/layers
- inference with 1000 iterations
- $\mathbf{\Phi} \in \{0,1\}^{248\times 961}$, $k = 31$
- $\mathcal{X} = \{s = 80 \text{ and } x_i^{\text{non-zero}} \sim \text{Beta iid}\}$
- NMAE$= \frac{\|x-\hat{x}\|_1}{\|x\|_1}$

Learning pooling matrices (k ones per row) and tuning viral load recovery

- projected subgradient descent for NNLAD can be unfolded
- slow convergence (not relevant for testing)
- supervised training with 200 iterations/layers
- inference with 1000 iterations
- $\mathbf{\Phi} \in \{0,1\}^{248\times 961}$, $k = 31$
- $\mathcal{X} = \{s = 80 \text{ and } x_i^{\text{non-zero}} \sim \text{Beta iid}\}$
- NMAE$= \frac{\|x-\hat{x}\|_1}{\|x\|_1}$



- ☹ optimal measurement matrices can not be constructed analytically, only random method
- ☹ classical algorithms well-understood but too slow and generic for realtime applications

☹ optimal measurement matrices can not be constructed analytically, only random method

☹ classical algorithms well-understood but too slow and generic for realtime applications

algorithm unfolding allows backprop and data-driven optimization of Φ

☺ overcome handcrafted hyper-parameter tuning

☺ learn better and constrained random models for Φ $\Rightarrow$ **derandomization**

"learn to sense and to recover"

☹ optimal measurement matrices can not be constructed analytically, only random method

☹ classical algorithms well-understood but too slow and generic for realtime applications

algorithm unfolding allows backprop and data-driven optimization of Φ

☺ overcome handcrafted hyper-parameter tuning

☺ learn better and constrained random models for $\Phi \Rightarrow$ **derandomization**

"learn to sense and to recover"

☺ examples in single-pixel/detector imaging and pooling

Thank you!

[1] E. J. Gumbel. *Statistical theory of extreme values and some practical applications: a series of lectures*. Vol. 33. US Government Printing Office, 1954.

[2] E. Jang, S. Gu, and B. Poole. "Categorical reparameterization with gumbel-softmax". *arXiv preprint arXiv:1611.01144* (2016).

[3] C. J. Maddison, A. Mnih, and Y. W. Teh. "The concrete distribution: A continuous relaxation of discrete random variables". *arXiv preprint arXiv:1611.00712* (2016).

[4] R. Kueng and P. Jung. "Robust Nonnegative Sparse Recovery and the Nullspace Property of 0/1 Measurements". *IEEE Transactions on Information Theory* 64.2 (Feb. 2018), 689–703.

[5] P. Jung, R. Kueng, and D. G. Mixon. "Derandomizing Compressed Sensing With Combinatorial Design". *Frontiers in Applied Mathematics and Statistics* 5 (June 2019). eprint: 1812.08130.

[6] H. B. Petersen, B. Bah, and P. Jung. "Practical high-throughput, non-adaptive and noise-robust SARS-CoV-2 testing". *arXiv preprint arXiv:2007.09171* (2020).

[7] F. Behrens, J. Sauder, and P. Jung. "Neurally Augmented ALISTA". *International Conference on Learning Representations (ICLR)* (2021).

[8] H. B. Petersen, B. Bah, and P. Jung. "Efficient Tuning-Free l1-Regression of Nonnegative Compressible Signals". *Frontiers in Applied Mathematics and Statistics* 7 (June 2021).

[9] B. Bah, H. B. Petersen, and P. Jung. "Compressed sensing-based SARS-CoV-2 pool testing". *Notices of the American Mathematical Society* 2 (2024).