ADMM-Net: An Algorithm Unrolling Approach For Network Resource Allocation

Zhonglin Xie

Peking University

July 31, 2021

Outline

What is L2O?

- Classic optimizers are manually designed, they usually have few or no tuning parameters
- Learned optimizers are trained in an L2O framework over a set of similar optimizees (called a task distribution) and designed to solve unseen optimizees from the same distribution

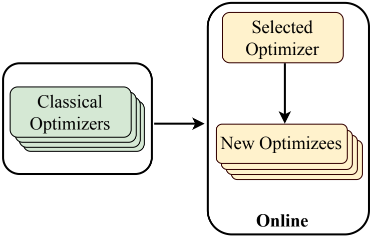


Figure 1: Classic Optimizer

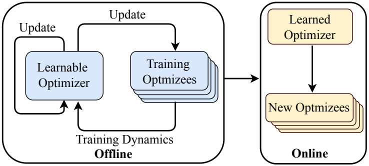
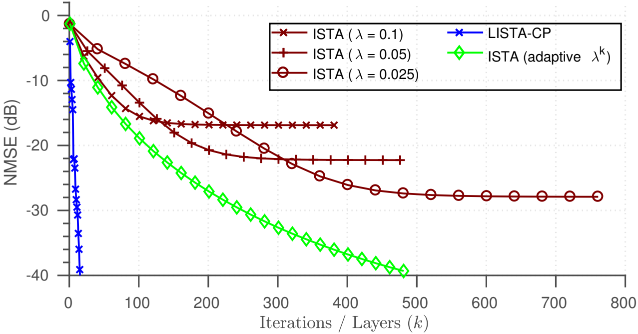


Figure 2: Learned Optimizer by L2O

Why L2O?

- An optimizer learned by L2O is much faster than classic methods
- The learned optimizer may also return a higher-quality solution to a difficult task than classic methods



Why Theory?

- We need an interpretable and reliable model with guaranteed worst performance
- Theory leads to an eifficient model with smaller size and less computational complexity (both in training and in testing)
- Heuristic method is cheap

Sparse Coding

- A classical problem in source coding, signal reconstruction, pattern recognition and feature selection
- There is an unknown sparse vector $x^* = [x_1^*, \cdots, x_M^*]^\top \in \mathbb{R}^M$. We have its noisy linear measurements:

$$b = \sum_{m=1}^{M} d_m x_m^* + \varepsilon = Dx^* + \varepsilon$$

  where $b \in \mathbb{R}^N, D = [d_1, \cdots, d_M] \in \mathbb{R}^{N \times M}$ is the dictionary, and $\varepsilon \in \mathbb{R}^N$ is additive Gaussian white noise
- Normalized dictionary: $\|d_m\|_2 = \|D_{:,m}\|_2 = 1, m = 1, 2, \cdots, M$
- Under-determined system: $N \ll M$
- Reconstruct x^* using a sparse linear combination of d_m
- Expensive inference algorithm prohibits real-time applications

Problem Formulation

$$\min_x \frac{1}{2}\|b - Dx\|_2^2 + \lambda\|x\|_1, \text{ where } b = Dx^* + \varepsilon$$

- A popular approach for sparse coding
- x^* can be recovered faithfully when it is sufficiently sparse
- Iterative Shrinkage Thresholding Algorithm (ISTA):

$$x^{k+1} = \eta_{\lambda/L}(x^k + \frac{1}{L}D^\top(b - Dx^k)), \quad k = 0, 1, 2, \ldots$$

where $\eta_\theta(x) = \operatorname{sign}(x)\max(0, |x| - \theta)$ and L is usually taken as the largest eigenvalue of $D^\top D$, λ is a hyper parameter

LISTA

- Let $W_1 = \frac{1}{L}D^\top$, $W_2 = I - \frac{1}{L}D^\top D$, $\theta = \frac{1}{L}\lambda$. ISTA can be written as
$$x^{k+1} = \eta_\theta(W_1 b + W_2 x^k)$$
- ISTA can be recognized as a Recurrent Neural Network (RNN)
- Unrolling the RNN and truncating it into K iterations:
$$x^{k+1} = \eta_{\theta^k}(W_1^k b + W_2^k x^k), \quad k = 0, 1, \cdots, K-1,$$
leads to a K-layer feed-forward neural network named Learned ISTA (LISTA) with trainable weights $\Theta = \{W_1^k, W_2^k, \theta^k\}_{k=1}^K$.

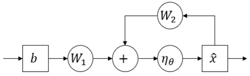


Figure 3: RNN Structure of ISTA

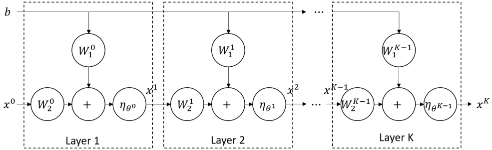


Figure 4: Unrolled Learned ISTA Network

LISTA with Coupling Weights (LISTA-CP)

We only parameterize the first D inside η, then get the LISTA with CouPling weights:

$$x^{k+1} = \eta_{\theta^k}(x^k + (W^k)^\top (b - Dx^k)),$$

where the trainable weights are reduced to $\Theta = \{W^k, \theta^k\}_{k=1}^K$.
Generalized Mutual Coherence:

$$\tilde{\mu}(D) = \inf_{\substack{W \in \mathbb{R}^{N \times M} \\ W_{:,i}^\top D_{:,i} = 1}} \left\{ \max_{\substack{i \neq j \\ 1 \leq i,j \leq M}} W_{:,i}^\top D_{:,j} \right\}$$

- LP: minimizing a piece-wise linear function with linear constraints
- Feasible, and

  $$0 \leq \tilde{\mu}(D) \leq \max_{\substack{i \neq j \\ 1 \leq i,j \leq M}} D_i^\top D_j.$$

  $\tilde{\mu}$ is bounded, exists optimal solution
- Define $\mathcal{W}(D) = \left\{ W \in \mathbb{R}^{N \times M} : W \text{ attains the infimum} \right\}$

Analytic LISTA: Less Parameters To Learn

- Tied LISTA (TiLISTA):
$$x^{k+1} = \eta_{\theta^k}\left(x^k - \gamma^k W^\top \left(Dx^k - b\right)\right),$$
where $\Theta = W \cup \{\gamma^k, \theta^k\}_{k=1}^K$ are trainable weights
- Following above theorem, we compute $\tilde{W}$ by solving
$$\tilde{W} \in \operatorname*{arg\,min}_{W \in \mathbb{R}^{N \times M}} \left\| W^\top D \right\|_F^2, \quad \text{s.t. } \left(W_{:,m}\right)^\top D_{:,m} = 1, \forall m$$
- We set $W^k = \gamma^k \tilde{W}$, and propose Analytic LISTA (ALISTA):
$$x^{k+1} = \eta_{\theta^k}(x^k - \gamma^k \tilde{W}^\top (Dx^k - b)),$$
where $\Theta = \{\gamma^k, \theta^k\}_{k=1}^K$ are parameters to train

Table 1: Summary: variants of LISTA and the number of parameters to learn.

LISTA	LISTA-CP	TiLISTA	ALISTA
$O\left(KM^2 + K + MN\right)$	$O(KNM + K)$	$O(NM + K)$	$O(K)$

Robust ALISTA to Model Perturbation (Meta-Net)

- Many applications, such as often found in surveillance video scenarios (Zhao et al., 2011; Han et al., 2013), can be formulated as sparse coding models whose dictionaries are subject to small dynamic perturbations (e.g, slowly varied over time)

$$\mathbf{D} : \mathbf{D} = \mathbf{D} + \varepsilon_D, \text{ where } \varepsilon_D \text{ is some small stochastic perturbation}$$

- Sample a perturbed dictionary $\tilde{\mathbf{D}}$. Sample $\mathbf{x}$ and ε to generate $\mathbf{b}$ w.r.t. $\tilde{\mathbf{D}}$. Apply Stage 1 of ALISTA w.r.t. $\tilde{\mathbf{D}}$ and obtain $\tilde{\mathbf{W}}$
- Instead of an iterative algorithm, we use a neural network that unfolds that algorithm (Meta-Net) to produce $\tilde{\mathbf{W}}$. Apply Stage 2 of ALISTA w.r.t. $\tilde{\mathbf{W}}, \mathbf{D}, \mathbf{x}$, and $\mathbf{b}$ to obtain $\{\gamma^k, \theta^k\}_k$
- $\tilde{\mathbf{D}}$ becomes the data for training the Meta-Net that generates $\tilde{\mathbf{W}}$
- This neural network is faster to apply than the iterative method

Training Process

- For each model, x^K depends on Θ, b, x^0. Denote x^K as $x^K(\Theta, b, x^0)$
- Given the distribution of b, x^*, the optimization problem is

$$\min_{\Theta} \mathbb{E}_{(b,x^*)} \|x^K(\Theta, b, x^0) - x^*\|_2^2.$$

Stochastic gradient descent (SGD) can be applied to solve this minimization problem. The gradient w.r.t. x^K on Θ are obtained with the chain rule

Trick: Layer-wise Training

- Denote $\Theta^\tau = \{(W_1^k, W_2^k, \theta^k)\}_{k=0}^{\tau}$ all the weights in the τ -th and all the previous layers
- Learning multiplier $c(\cdot)$ initialized as 1 to each weight
- Initial learning rate α_0 and two decayed learning rates α_1, α_2. In real training, we have $\alpha_1 = 0.2\alpha_0, \alpha_2 = 0.02\alpha_0$
- Train $(W_1^\tau, W_2^\tau, \theta^\tau)$ the initial learning rate α_0
- Train $\Theta^\tau = \Theta^{\tau-1} \cup (W_1^\tau, W_2^\tau, \theta^\tau)$ with the learning rates α_1 and α_2
- Multiply a decaying rate γ (set to 0.3 in experiments) to each weight in Θ^τ

Code for Layer-wise Training

```
loss_ = tf.nn.l2_loss (xhs_ [t+1] - x_)

var_list = tuple([var for var in model.vars_in_layer[t] if
    var not in train_vars])

op_ = tf.train.AdamOptimizer(init_lr).minimize(loss_,
    var_list=var_list)
...
for var in var_list:
  train_vars.append (var)

# Train all variables in current and former layers with
    decayed
for lr in lrs:
  op_ = tf.train.AdamOptimizer(lr_multiplier*lr).minimize(
    loss_,var_list=train_vars)
# decay learning rates for trained variables
for var in train_vars:
  lr_multiplier [var.op.name] *= decay_rate
```

Neccessary Condition for Convergence

- Assumption: $b = Dx^*, x^0 = 0$ and

$$x^* \in \mathcal{X}(B,s) \triangleq \{x^* \mid |x_i^*| \leq B, \forall i, \ \|x^*\|_0 \leq s\}$$

- x^k depends on $\{W_1^\tau, W_2^\tau, \theta^\tau\}_{\tau=0}^{k-1}, b, x^0$. Using $b = Dx^*, x^0 = 0, x^k$ can be represented as

$$x^k(\{W_1^\tau, W_2^\tau, \theta^\tau\}_{\tau=0}^{k-1}, x^*)$$

Theorem (Neccessary Condition for Convergence of LISTA)

If $x^k(\{W_1^\tau, W_2^\tau, \theta^\tau\}_{\tau=0}^{k-1}, x^*) \to x^*$ uniformly for $x^* \in \mathcal{X}(B,s)$ as $k \to \infty$, and $\|W_2^k\|_2 \leq B_W, \forall k$, where B_W is a positive constant, we have

$$\theta^k \to 0, \ W_2^k - (I - W_1^k D) \to 0, \ \textit{as } k \to \infty$$

Proof of $\theta^k \to 0$

- Since $x^k \to x^*$ uniformly, there exists $K_1, \forall k \geq K_1, |x_i^k - x_i^*| < \frac{B}{10}$
- Denote

$$\mathcal{X}(\tilde{B}, B, s) \triangleq \{x^* \mid \tilde{B} \leq |x_i^*| \leq B, \forall i \in \operatorname{supp}(x^*),\ \|x^*\|_0 \leq s\}$$

- Above inequality holds for any $x^* \in \mathcal{X}(B/10, B, s)$, we have

$$\operatorname{sign}(x^k) = \operatorname{sign}(x^*),\ \forall k \geq K_1$$

- Let $S = \operatorname{supp}(x^*)$, consider the support set elements

$$\begin{aligned} x_S^{k+1} =& \eta_{\theta^k}(W_2^k(S,S)x_S^k + W_1^k(S,:)b) \\ =& W_2^k(S,S)x_S^k + W_1^k(S,:)b - \theta^k \operatorname{sign}(x_S^*) \end{aligned}$$

Proof of $\theta^k \to 0$

- $\forall \varepsilon > 0$, there exists K_2, such that $\forall k \geq K_2$, $\|x^k - x^*\|_2 \leq \varepsilon$
- Suppose $k \geq \max\{K_1, K_2\}$, and $x^k = x^* + \xi_1, x^{k+1} = x^* + \xi_2$, then

$$x_S^{k+1} = W_2^k(S,S)x_S^k + W_1^k(S,:)b - \theta^k \ \mathrm{sign}(x_S^*)$$
$$\Leftrightarrow x_S^* + \xi_2 = W_2^k(S,S)(x_S^* + \xi_1) + W_1^k(S,:)b - \theta^k \ \mathrm{sign}(x_S^*)$$

- Denote $\xi = W_2^k(S,S)\xi_1 - \xi_2$, we have $\|\xi\|_2 \leq (1 + B_W)\varepsilon$ and

$$(I - W_2^k(S,S) - W_1^k D(S,S))x_S^* = \theta^k \ \mathrm{sign}(x_S^*) - \xi$$

- Take $x^* \in \mathcal{X}(B/10, B/2, s)$, above formula holds for $2x^*$

$$(I - W_2^k(S,S) - W_1^k D(S,S))2x_S^* = \theta^k \ \mathrm{sign}(x_S^*) - \xi'$$

- Subtracting the above two formulas yields

$$\theta^k \mathrm{sign}(x_S^*) = 2\xi - \xi' \Rightarrow \theta^k \leq \frac{3(1 + B_W)}{\sqrt{|S|}}\varepsilon \Rightarrow \theta^k \to 0$$

Proof of $W_2^k - (I - W_1^k D) \to 0$

- By optimality condition

$$x_S^{k+1} \in W_2^k(S,:)x^k + W_1^k D(S,S)x_S^* - \theta^k \partial \ell_1(x_S^{k+1}),$$

where $\partial \ell_1(x)$ is the sub-gradient of $\|x\|_1$

- x^k converging uniformly implies, for any $\varepsilon > 0, x^* \in \mathcal{X}(B,s)$, there exists K_3, such that $\forall k \geq K_3, \|x^k - x^*\|_2 \leq \varepsilon$
- Suppose $x^k = x^* + \xi_3, x^{k+1} = x^* + \xi_4$, above formula equals to

$$\Big(I - W_2^k(S,S) - W_1^k D(S,S)\Big)x_S^* \in W_2^k(S,:)\xi_3 - (\xi_4)_S - \theta^k \partial \ell_1(x_S^{k+1})$$

- By $\|\partial \ell_1(x_S^{k+1})\|_2 \leq \sqrt{|S|}$

$$\Big\|\Big(I - W_2^k(S,S) - W_1^k D(S,S)\Big)x_S^*\Big\|_2 \leq \|W_2^k\|_2 \varepsilon + \varepsilon + \theta^k \sqrt{|S|} \to 0$$

- By the arbitrariness of the $x^* \in \mathcal{X}(B,s) \Rightarrow W_2^k - (I - W_1^k D) \to 0$

Recovery Error Upper Bound

Theorem (Recovery Error Upper Bound of LISTA-CP)

Take any $x^* \in \mathcal{X}(B, s)$, *any* $W \in \mathcal{W}(D)$, *any* $\gamma^k \in (0, \frac{2}{2\tilde{\mu}s - \tilde{\mu} + 1})$. *Using them, define the parameters* $\{W^k, \theta^k\}$

$$W^k = \gamma^k W, \quad \theta^k = \gamma^k \tilde{\mu}(D) \sup_{x^* \in \mathcal{X}(B,s)} \{\|x^k(x^*) - x^*\|_1\}$$

while the sequence $\{x^k(x^*)\}_{k=1}^{\infty}$ *is generated by LISTA-CP using the above parameters and* $x^0 = 0$ *(Note that each* $x^k(x^*)$ *depends only on* $\theta^{k-1}, \theta^{k-2}, \ldots$ *and defines* θ^k*). Let* $s < (1 + 1/\tilde{\mu})/2$. *We have*

$$\operatorname{supp}(x^k(x^*)) \subset S, \quad \|x^k(x^*) - x^*\|_2 \leq sB \exp(-\sum_{\tau=0}^{k-1} c^\tau), \quad k = 1, 2, \ldots$$

where $S = \operatorname{supp}(x^*)$ *and* $c^k = -\log((2\tilde{\mu}s - \tilde{\mu})\gamma^k + |1 - \gamma^k|) > 0.$

Parameters Selection with No False Positives

- We have the neccessary condition:

$$x^k \to x^* \Rightarrow \frac{\|x_S^k\|_2}{\|x^k\|_2} \to \frac{\|x_S^*\|_2}{\|x^*\|_2}$$

- Does there exist θ^k, γ^k, such that for any $k, \operatorname{supp}(x^k) \subset S$?
- Yes. Assuming $\operatorname{supp}(x^k) \subset S$, for any $i \notin S$, we have

$$x_i^{k+1} = \eta_{\theta^k}(-\gamma^k \sum_{j \in S} W_{:,j}^\top (Dx^k - b))$$

- Note that $\eta_\theta(x) = \operatorname{sign}(x) \max(|x| - \theta, 0)$. When

$$\theta^k = \gamma^k \tilde{\mu}(D) \sup_{x^* \in \mathcal{X}(B,s)} \{\|x^k(x^*) - x^*\|_1\} \geq \gamma^k |\sum_{j \in S} W_{:,j}^\top (Dx^k - b)|,$$

we have $x_i^{k+1} = 0$.

Proof of Recovery Error Upper Bound

- Take arbitrary $x^* \in \mathcal{X}(B,s)$. For all $i \in S$, by optimality condition, we obtain

$$x_i^{k+1} \in x_i^k - \gamma^k W_{:,i}^\top D_{:,S}(x_S^k - x_S^*) - \theta^k \partial \ell_1(x_i^{k+1})$$

where $\partial \ell_1(x)$ is the sub-gradient of $|x|, x \in \mathbb{R}:$

$$\partial \ell_1(x) = \begin{cases} \{\mathrm{sign}(x)\} & \text{if } x \neq 0 \\ [-1,1] & \text{if } x = 0 \end{cases}$$

- The choice of $W \in \mathcal{W}(D)$ gives $W_{:,i}^\top D_{:,i} = 1$. Thus,

$$\begin{aligned} & x_i^k - \gamma^k W_{:,i}^\top D_{:,S}(x_S^k - x_S^*) \\ = & x_i^k - \gamma^k \sum_{j \in S, j \neq i} W_{:,i}^\top D_{:,j}(x_j^k - x_j^*) - \gamma^k (x_i^k - x_i^*) \\ = & x_i^* - \gamma^k \sum_{j \in S, j \neq i} W_{:,i}^\top D_{:,j}(x_j^k - x_j^*) + (1 - \gamma^k)(x_i^k - x_i^*) \end{aligned}$$

Proof of Recovery Error Upper Bound

- For all $i \in S$

$$x_i^{k+1} \in x_i^* - \gamma^k \sum_{j \in S, j \neq i} W_{:,i}^\top D_{:,j}(x_j^k - x_j^*) + (1-\gamma^k)(x_i^k - x_i^*) - \theta^k \partial \ell_1(x_i^{k+1})$$

- Thus

$$\begin{aligned} |x_i^{k+1} - x_i^*| &\leq \sum_{j \in S, j \neq i} \gamma^k |W_{:,i}^\top D_{:,j}| |x_j^k - x_j^*| + \theta^k + |1-\gamma^k| |x_i^k - x_i^*| \\ &\leq \tilde{\mu}\gamma^k \sum_{j \in S, j \neq i} |x_j^k - x_j^*| + \theta^k + |1-\gamma^k| |x_i^k - x_i^*| \end{aligned}$$

- Note that $\|x^k - x^*\|_1 = \|x_S^k - x_S^*\|_1$, summing $i \in S$ yields

$$\begin{aligned} \|x^{k+1} - x^*\|_1 \leq & (|S|-1)\tilde{\mu}\gamma^k \|x^k - x^*\|_1 + |S|\theta^k + |1-\gamma^k| \|x^k - x^*\|_1 \\ = & ((|S|-1)\tilde{\mu}\gamma^k + |1-\gamma^k|)\|x^k - x^*\|_1 + |S|\theta^k \end{aligned}$$

Proof of Recovery Error Upper Bound

- The assumption $s < (1+1/\tilde{\mu})/2$ gives $2\tilde{\mu}s - \tilde{\mu} < 1$. If $0 < \gamma^k \le 1$, we have $c^k > 0$. If $1 < \gamma^k < 2/(1+2\tilde{\mu}s-\tilde{\mu})$, we have

$$(2\tilde{\mu}s-\tilde{\mu})\gamma^k + \left|1-\gamma^k\right| = (2\tilde{\mu}s-\tilde{\mu})\gamma^k + \gamma^k - 1 < 1,$$

  which implies $c^k = -\log((2\tilde{\mu}s-\tilde{\mu})\gamma^k + |1-\gamma^k|) > 0$

- Taking supremum of the last inequality over $x^* \in \mathcal{X}(B,s)$, by $|S| \le s$ and $\theta^k = \gamma^k\tilde{\mu}\sup_{x^*}\|x^k - x^*\|_1$,

$$\begin{aligned}
\sup_{x^*}\|x^{k+1}-x^*\|_1 &\le ((2\tilde{\mu}s-\tilde{\mu})\gamma^k + |1-\gamma^k|)\sup_{x^*}\|x^k-x^*\|_1 \\
&\le \exp(-\sum_{\tau=0}^{k} c^\tau)\sup_{x^*}\|x^0-x^*\|_1 \\
&\le sB\exp(-\sum_{\tau=0}^{k} c^\tau)
\end{aligned}$$

Numerical Results

- Settings: $N = 250, M = 500$ and $D_{i,j} \sim \mathcal{N}(0, \frac{1}{N})$ with $\|D_{:,j}\|_2 = 1$
- Set the number of truncated layers $K = 16$. Training process:

$$\min_{\Theta} \mathbb{E}_{x^*} \|x^K(\Theta) - x^*\|_2$$

- Θ is learnable parameters and is different in different models

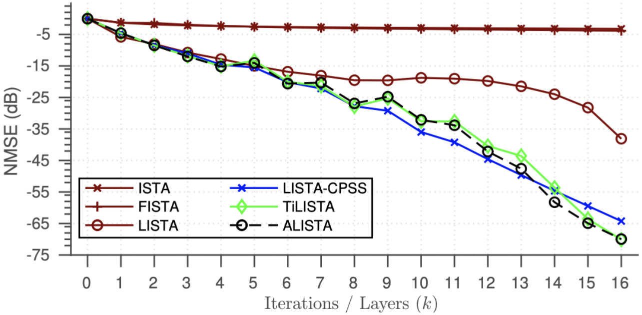


Figure 5: $\text{NMSE} = 10 \log_{10} \left(\frac{\mathbb{E}\|x^K(\Theta) - x^*\|_2}{\mathbb{E}\|x^*\|_2} \right)$

Validation of Neccessary Condition for Convergence

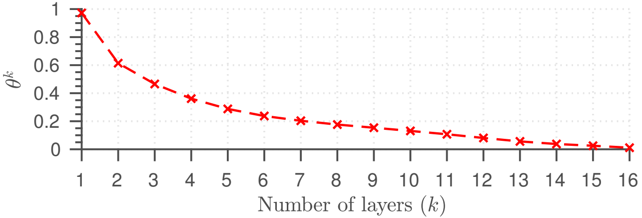


Figure 6: $\theta^k \to 0$

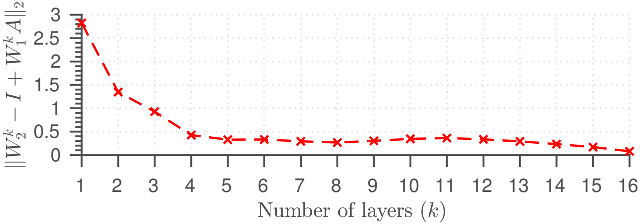


Figure 7: $W_2^k - (I - W_1^k D) \to 0$

Validation of Theorem

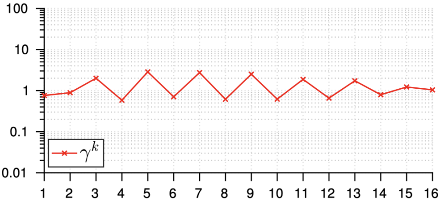


Figure 8: $\gamma^k \in (0, \frac{2}{2\tilde{\mu}s - \tilde{\mu} + 1})$

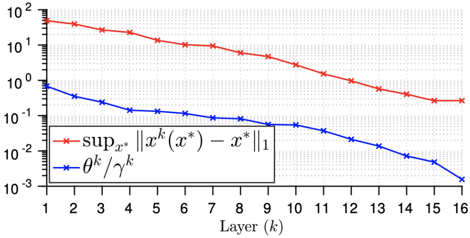


Figure 9: $\theta^k = \gamma^k \tilde{\mu} \sup\{\|x^k(x^*) - x^*\|_1\}$

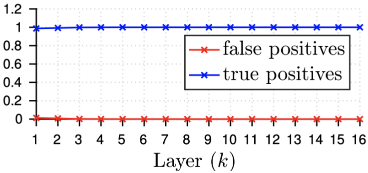


Figure 10: true positives $= \frac{\mathbb{E}\|x_S^k(x^*)\|_2}{\mathbb{E}\|x^k(x^*)\|_2}$, false positives $= \frac{\mathbb{E}\|x_{S^c}^k(x^*)\|_2}{\mathbb{E}\|x^k(x^*)\|_2}$

CS-MRI

- Magnetic Resonance Imaging (MRI) is a non-invasive imaging technique for clinical diagnosis
- Compressive sensing MRI (CS-MRI) methods first sample data, then reconstruct image using compressive sensing theory
- Challenging to choose an optimal image transform domain and the corresponding sparse regularization
- Alternating Direction Method of Multipliers (ADMM) is efficient but it is not trivial to determine the optimal parameters

General CS-MRI Model

- Assume $x \in \mathbb{C}^N$ is an MRI image to be reconstructed
- $y \in \mathbb{C}^{N'}(N' < N)$ is the under-sampled k-space data
- The reconstructed image can be estimated by solving:

$$\hat{x} = \arg\min_{x}\{\frac{1}{2}\|Ax - y\|_2^2 + \sum_{l=1}^{L} \lambda_l g(D_l x)\},$$

where $A = PF \in \mathbb{R}^{N' \times N}$ is a measurement matrix, $P \in \mathbb{R}^{N' \times N}$ is a under-sampling matrix, and F is a Fourier transform. D_l denotes a transform matrix for a filtering operation. $g(\cdot)$ is a regularization function. λ_l is a regularization parameter

ADMM Algorithm

- Introduce auxiliary variables $z = \{z_1, z_2, \cdots, z_L\}$:

$$\min_{x,z} \frac{1}{2}\|Ax - y\|_2^2 + \sum_{l=1}^{L} \lambda_l g(z_l) \quad \text{s.t. } z_l = D_l x, \quad l = 1, 2, \cdots, L$$

- Augmented Lagrangian function:

$$\begin{aligned} L_\rho(x, z, \alpha) = \frac{1}{2}\|Ax - y\|_2^2 + \sum_{l=1}^{L} \lambda_l g(z_l) - \sum_{l=1}^{L} \langle \alpha_l, z_l - D_l x \rangle \\ + \sum_{l=1}^{L} \frac{\rho_l}{2}\|z_l - D_l x\|_2^2 \end{aligned}$$

where $\alpha = \{\alpha_l\}$ are Lagrangian multipliers and $\rho = \{\rho_l\}$ are penalty parameters

ADMM Algorithm

- Alternatively optimizes $\{x, z, \alpha\}$ and substitute $A = PF, \beta_l = \frac{\alpha_l}{\rho_l}$:

$$\begin{cases} x^n = F^\top G^{-1}[P^\top y + \sum_{l=1}^{L} \rho_l F D_l^\top (z_l^{n-1} - \beta_l^{n-1})] \\ z_l^n = S(D_l x^n + \beta_l^{n-1}; \lambda_l/\rho_l) \\ \beta_l^n = \beta_l^{n-1} + \eta_l (D_l x^n - z_l^n) \end{cases}$$

  where $G = P^\top P + \sum_{l=1}^{L} \rho_l F D_l^\top D_l F^\top$, $S(\cdot)$ is a nonlinear shrinkage function, η_l is an update rate
- x^n can be efficiently computed by Fast Fourier Transform
- Needs to run dozens of iterations to get a satisfactory result
- Challenging to choose the transform D_l and shrinkage function $S(\cdot)$ for general regularization function $g(\cdot)$
- Not trivial to tune the parameters ρ_l and η_l for different data

Data Flow Graph for the ADMM Algorithm

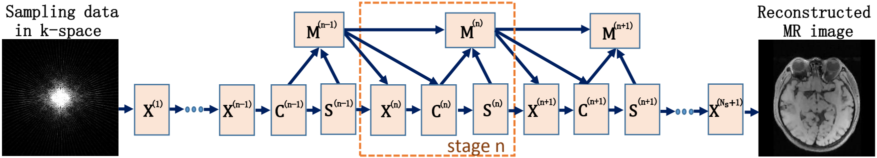


Figure 11: The data flow graph for the ADMM. This graph consists of four types of nodes: reconstruction ($\mathbf{X}$), convolution ($\mathbf{C}$), non-linear transform ($\mathbf{Z}$), and multiplier update ($\mathbf{M}$).

ADMM-Net

- ADMM-Net is defined over the data flow graph
- Reconstruction layer $\mathbf{X}^n$: Substituting D_l, ρ_l with H_l^n, ρ_l^n, we get

$$x^n = F^\top (P^\top P + \sum_{l=1}^{L} \rho_l^n F (H_l^n)^\top H_l^n F^\top)^{-1} [P^\top y \\ + \sum_{l=1}^{L} \rho_l^n F (H_l^n)^\top (z_l^{n-1} - \beta_l^{n-1})]$$

  where H_l^n is the l-th learnable filter, ρ_l^n is the l-th learnable penalty parameter, and y is the input under-sampled data
- Convolution layer $\mathbf{C}^n$:

$$c_l^n = D_l^n x^n$$

  where D_l^n is a learnable filter matrix in stage n. Different from the original ADMM, we do not constrain the filters D_l^n and H_l^n to be the same to increase the network capacity

ADMM-Net

- Nonlinear transform layer $\mathbf{Z}^n$: Use piecewise linear function to replace the shrinkage function $S(\cdot)$. Given c_l^n and β_l^{n-1}:

$$z_l^n = S_{PLF}(c_l^n + \beta_l^{n-1}; \{p_i, q_{l,i}^n\}_{i=1}^{N_c}),$$

where $S_{PLF}(\cdot)$ is determined by a set of control points $\{p_i, q_{l,i}^n\}_{i=1}^{N_c}$

$$S_{PLF}(a; \{p_i, q_{l,i}^n\}_{i=1}^{N_c}) = \begin{cases} a + q_{l,1}^n - p_1, & a < p_1, \\ a + q_{l,N_c}^n - p_{N_c}, & a > p_{N_c} \\ q_{l,k}^n + \frac{(a-p_k)(q_{l,k+1}^n - q_{l,k}^n)}{p_{k+1}-p_k}, & p_1 \leq a \leq p_{N_c} \end{cases}$$

where $k = \lfloor \frac{a-p_1}{p_2-p_1} \rfloor$, $\{p_i\}_{i=1}^{N_c}$ are predefined positions uniformly located within $[-1, 1]$, and $\{q_{l,i}^n\}_{i=1}^{N_c}$ are the values at these positions for l-th filter in n-th stage

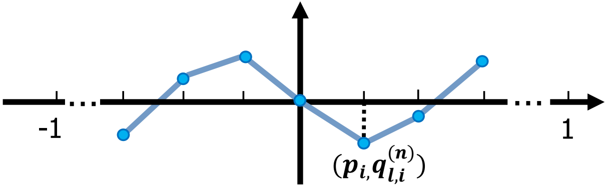


Figure 12: Illustration of a piecewise linear function $S_{PLF}(\cdot; \{p_i, q_{l,i}^n\}_{i=1}^{N_c})$.

- Multiplier update layer $\mathbf{M}^n$:

$$\beta_l^n = \beta_l^{n-1} + \eta_l^n (c_l^n - z_l^n)$$

  where η_l^n are learnable parameters.
- Network Parameters: H_l^n and ρ_l^n in reconstruction layer, filters D_l^n in convolution layer, $\{q_{l,i}^n\}_{i=1}^{N_c}$ in nonlinear transform layer, η_l^n in multiplier update layer, where $l = 1, 2, \cdots, L$ and $n = 1, 2, \cdots, N_s$

Network Training

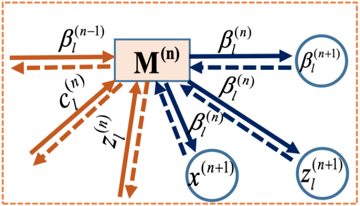
Given the training data Γ, the loss function is:

$$E(\Theta) = \frac{1}{|\Gamma|} \sum_{(y,x^*)\in\Gamma} \frac{\|\hat{x}(y,\Theta) - x^*\|_2}{\|x^*\|_2}$$

where $\hat{x}(y,\Theta)$ is the network output based on network parameter Θ and under-sampled data y, $\Theta_l = \{(q^n_{l,i})^{N_c}_{i=1}, D^n_l, H^n_l, \rho^n_l, \eta^n_l\}^{N_s}_{n=1}, \Theta = \{\Theta_l\}^L_{l=1}$

We learn the parameters by minimizing the loss w.r.t. Θ using L-BFGS

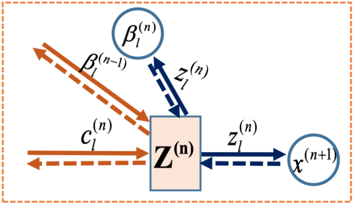
Multiplier Update Layer



(a) Multiplier update layer

- Three sets of inputs: $\{\beta_l^{n-1}\}, \{c_l^n\}$ and $\{z_l^n\}$
- Its output $\{\beta_l^n\}$ is the input to compute $\{\beta_l^{n+1}\}, \{z_l^{n+1}\}$ and x^{n+1}
- The parameters of this layer are $\eta_l^n, l = 1, \cdots, L$
- The gradients of loss w.r.t. the parameters can be computed as: $\frac{\partial E}{\partial \eta_l^n} = \frac{\partial E}{\partial \beta_l^n} \frac{\partial \beta_l^n}{\partial \eta_l^n}$, where $\frac{\partial E}{\partial \beta_l^n} = \frac{\partial E}{\partial \beta_l^{n+1}} \frac{\partial \beta_l^{n+1}}{\partial \beta_l^n} + \frac{\partial E}{\partial z_l^{n+1}} \frac{\partial z_l^{n+1}}{\partial \beta_l^n} + \frac{\partial E}{\partial x^{n+1}} \frac{\partial x^{n+1}}{\partial \beta_l^n}$

Nonlinear Transform Layer

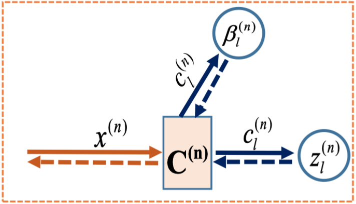


(b) Non-linear transform layer

- Two sets of inputs:$\{\beta_l^{n-1}\}, \{c_l^n\}$
- Its output $\{z_l^n\}$ is the input for computing $\{\beta_l^n\}$ and x^{n+1}
- The parameters of this layers are $\{q_{l,i}^n\}_{i=1}^{N_c}, l = 1, \cdots, L$
- The gradient of loss w.r.t. parameters can be computed as

$$\frac{\partial E}{\partial q_{l,i}^n} = \frac{\partial E}{\partial z_l^n} \frac{\partial z_l^n}{\partial q_{l,i}^n}, \text{ where } \frac{\partial E}{\partial z_l^n} = \frac{\partial E}{\partial \beta_l^n} \frac{\partial \beta_l^n}{\partial z_l^n} + \frac{\partial E}{\partial x^{n+1}} \frac{\partial x^{n+1}}{\partial z_l^n}$$

Convolution Layer



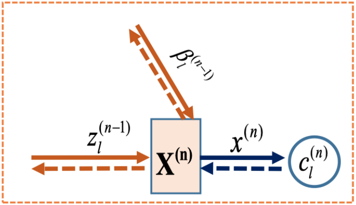
(c) Convolution layer

- The parameters of this layer are $D_l^n (l = 1, \cdots, L)$. We represent the filter by $D_l^n = \sum_{m=1}^{t} \omega_{l,m}^n B_m$, where B_m is a basis element, and $\{\omega_{l,m}^n\}$ is the set of filter coefficients to be learned
- The gradients of loss w.r.t. filter coefficients are computed as:

$$\frac{\partial E}{\partial \omega_{l,m}^n} = \frac{\partial E}{\partial c_l^n} \frac{\partial c_l^n}{\partial \omega_{l,m}^n}, \text{ where } \frac{\partial E}{\partial c_l^n} = \frac{\partial E}{\partial \beta_l^n} \frac{\partial \beta_l^n}{\partial c_l^n} + \frac{\partial E}{\partial z_l^n} \frac{\partial z_l^n}{\partial c_l^n}$$

- The gradient of layer output w.r.t. input is computed as $\frac{\partial c_l^n}{\partial x^n}$

Reconstruction Layer



(d) Reconstruction layer

- The parameters of this layer are $H_l^n, \rho_l^n (l = 1, \cdots, L)$
- Represent the filter by $H_l^n = \sum_{m=1}^{s} \gamma_{l,m}^n B_m$, $\{\gamma_{l,m}^n\}$ is learnable
- The gradients w.r.t. parameters: $\frac{\partial E}{\partial \gamma_{l,m}^n} = \frac{\partial E}{\partial x^n} \frac{\partial x^n}{\partial \gamma_{l,m}^n}, \frac{\partial E}{\partial \rho_l^n} = \frac{\partial E}{\partial x^n} \frac{\partial x^n}{\partial \rho_l^n}$

$$\text{where } \frac{\partial E}{\partial x^n} = \begin{cases} \frac{\partial E}{\partial c^n} \frac{\partial c^n}{\partial x^n}, & n \leq N_s \\ \frac{1}{|\Gamma|} \frac{(x^n - x^{gt})}{\sqrt{\|x^{gt}\|_2^2} \sqrt{\|x^{n - x^{gt}}\|_2^2}}, & \text{if } n = N_s + 1 \end{cases}$$

Result

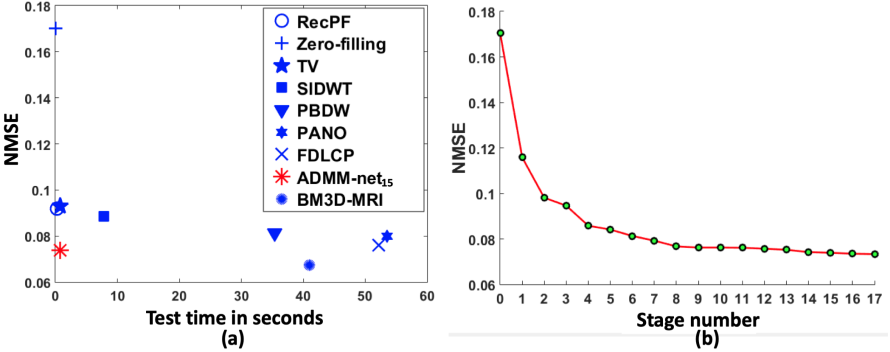
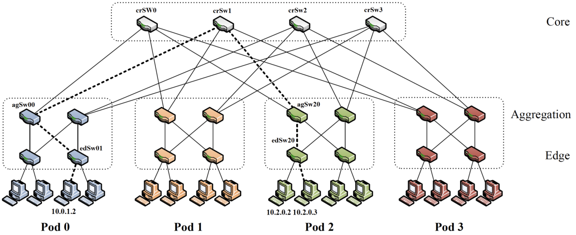


Figure 13: (a) Scatter plot of NMSEs and average test time for different methods; (b) The NMSEs of ADMM-Net using different number of stages (20% sampling ratio for brain data).

Background

- Wide area network (WAN): to transmit data over long distances
- There are K flows and the size of k-th flow is s_k; total available number of paths for flow k is P_k; link capacity $c_l, l = 1, 2, \cdots, L$. E.g. 4-ary Fat Tree topology



- Maximize one kind of utility functions among all flows, i.e. $\sum_{k=1}^{K} U_k(\|\boldsymbol{x_k}\|_1)$, where $\boldsymbol{x_k}$ is the rate allocation and path selection vector of flow k

$$\begin{aligned} \max \quad & \log(x_{1,1} + x_{1,2}) + \log(x_{2,1} + x_{2,2}), \\ \text{s.t.} \quad & x_{2,1} + x_{1,2} \leq 1, \cdots \end{aligned}$$

- K flows with the size s_k for the flow k
- P_k available paths for flow k
- L links with the link capacity c_l for the link l
- $\boldsymbol{x}_k = (x_{k,1}, \cdots, x_{k,P_k})^\top$ is the rate allocation vector of flow k
- Routing matrix: $\boldsymbol{R} = (\boldsymbol{R}_1, \boldsymbol{R}_2, \cdots, \boldsymbol{R}_K)$

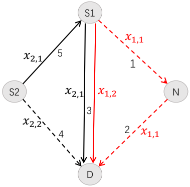


Figure 14: A network with two users and five links.

$$\boldsymbol{R}_1 = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \boldsymbol{R}_2 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Figure 15: Routing matrix.

Network Utility Maximization (NUM)

- Common choices of $U_k(\|\boldsymbol{x_k}\|_1)$: fairness $\log(\|\boldsymbol{x_k}\|_1)$ or delay $-\frac{s_k}{\|\boldsymbol{x_k}\|_1}$. We choose $U_k(\|\boldsymbol{x_k}\|_1) = \beta \log(\|\boldsymbol{x_k}\|_1) - s_k/\|\boldsymbol{x_k}\|_1$
- Network Utility Maximization (NUM) problem:

$$\begin{aligned} \max_{\boldsymbol{x}} \quad & \sum_k U_k(\|\boldsymbol{x_k}\|_1), \\ \text{s.t.} \quad & \boldsymbol{R}\boldsymbol{x} \leq \boldsymbol{c}, \\ & \boldsymbol{x} \geq 0, \end{aligned} \iff \begin{aligned} \max_{\boldsymbol{x},\boldsymbol{y}} \quad & \sum_k U_k(\|\boldsymbol{x_k}\|_1), \\ \text{s.t.} \quad & \boldsymbol{y} \leq \boldsymbol{c}, \\ & \boldsymbol{x} \geq 0, \\ & \boldsymbol{y} = \boldsymbol{R}\boldsymbol{x} \end{aligned}$$

where $\boldsymbol{R} = [\boldsymbol{R}_1, \boldsymbol{R}_2, \cdots, \boldsymbol{R}_K] \in \mathbb{R}^{L \times K}$ is the routing matrix (sparse), $\boldsymbol{x} = [\boldsymbol{x}_1; \boldsymbol{x}_2; \cdots; \boldsymbol{x}_K]$

- Augmented Lagrangian:

$$L_\rho(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = -\sum_{k=1}^{K} U_k(\|\boldsymbol{x}_k\|_1) - \boldsymbol{z}^\top(\boldsymbol{y} - \boldsymbol{R}\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{y} - \boldsymbol{R}\boldsymbol{x}\|_2^2$$

x-update

$$\boldsymbol{x}^j \leftarrow \arg\min_{\boldsymbol{x}} -\sum_{k=1}^{K} U_k(\|\boldsymbol{x}_k\|_1) - (\boldsymbol{z}^{j-1})^\top(\boldsymbol{y}^{j-1} - \boldsymbol{R}\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{y}^{j-1} - \boldsymbol{R}\boldsymbol{x}\|_2^2,$$

- Hard to solve, because the components of $\boldsymbol{x}$ are coupled
- Linearize the quadratic term and add a proximal term:

$$\begin{aligned}\boldsymbol{x}^j &= \arg\min_{\boldsymbol{x}} -\sum_{k=1}^{K} U_k(\|\boldsymbol{x}_k\|_1) - \rho\langle \boldsymbol{R}^\top \xi, \boldsymbol{x} - \boldsymbol{x}^{j-1}\rangle + \frac{\mu}{2}\|\boldsymbol{x} - \boldsymbol{x}^{j-1}\|_2^2 \\ &= \arg\min_{\boldsymbol{x}} -\sum_{k=1}^{K} U_k(\|\boldsymbol{x}_k\|_1) + \frac{\mu}{2}\|\boldsymbol{x} - \boldsymbol{x}^{j-1} - \frac{\rho}{\mu}\boldsymbol{R}^\top \xi^{j-1}\|_2^2,\end{aligned}$$

where $\xi^{j-1} = \boldsymbol{y}^{j-1} - \boldsymbol{R}\boldsymbol{x}^{j-1} - \frac{\boldsymbol{z}^{j-1}}{\rho}$

x-update

- Separable for different source. For k-th source:

$$\boldsymbol{x}_k^j = \arg\min_{\boldsymbol{x}_k} -U_k(\|\boldsymbol{x}_k\|_1) + \frac{\mu}{2}\|\boldsymbol{x}_k - \boldsymbol{\nu}_k^{j-1}\|_2^2,$$

where $\boldsymbol{\nu}^{j-1} = \boldsymbol{x}^{j-1} + \frac{\rho}{\mu}\boldsymbol{R}^\top(\boldsymbol{y}^{j-1} - \boldsymbol{R}\boldsymbol{x}^{j-1} - \frac{\boldsymbol{z}^{j-1}}{\rho})$

- The elements of $\boldsymbol{\nu}_k^{j-1} = (\nu_{k,1}^{j-1}, \nu_{k,2}^{j-1}, \cdots, \nu_{k,P_k}^{j-1})^\top$ are in descending order:

$$x_{k,i}^j = \max(0, \nu_{k,i}^{j-1} + \zeta_k), \text{ where } \mu i' \zeta_k = U_k'(\sum_{i=1}^{i'} \max(0, \nu_{k,i}^{j-1} + \zeta_k)),$$

i' is the maximal index: $U_k'(\sum_{i=1}^{i'} \max(0, \nu_{k,i}^{j-1} - \nu_{k,i'}^{j-1})) \geq -\mu\nu_{k,i'}^{j-1}$

- ζ_k can be found by solving

$$r_k - \frac{\beta}{\mu}\frac{1}{r_k} - \frac{s_k}{\mu K}\frac{1}{r_k^2} = \frac{\sum_{i=1}^{i'} \nu_{k,i}^{j-1}}{\mu}$$

where $r_k = \sum_{i=1}^{i'}(\nu_{k,i}^{j-1} + \zeta_k)$

x-update

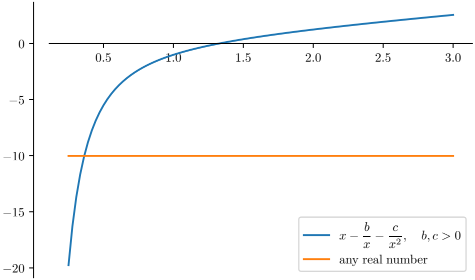


Figure 16: An illustration of finding ζ_k.

We denote the mapping between ν_k^{j-1} and x_k^j as

$$x_k^j = \mathcal{C}_k(\nu_k^{j-1})$$

y-update

$$\boldsymbol{y}^j \leftarrow \arg\min_{\boldsymbol{y} \leq \boldsymbol{c}} \frac{\rho}{2} \|\boldsymbol{y} - \boldsymbol{R}\boldsymbol{x}^j\|_2^2 - (\boldsymbol{z}^{j-1})^\top (\boldsymbol{y} - \boldsymbol{R}\boldsymbol{x}^j).$$

The solution is

$$\boldsymbol{y}^j = -\mathcal{P}_{\mathbb{R}_+^L}(\boldsymbol{c} - \boldsymbol{R}\boldsymbol{x}^j - \frac{\boldsymbol{z}^{j-1}}{\rho}) + \boldsymbol{c},$$

where $\mathcal{P}_{\mathbb{R}_+^L}$ is the Euclidean projection on $\mathbb{R}_+^L = \{(x_1, x_2, \cdots, x_L)^\top \mid x_i \geqslant 0,\ i = 1, 2, \cdots, L\}$

ADMM for NUM

$$
\begin{cases}
\boldsymbol{x}_k^j \leftarrow \mathcal{C}_k(\boldsymbol{\nu}_k^{j-1}), k = 1, \ldots, K, \\
\boldsymbol{y}^j \leftarrow -\mathcal{P}_{\mathbb{R}_+^L}(\boldsymbol{c} - \boldsymbol{R}\boldsymbol{x}^j - \dfrac{\boldsymbol{z}^{j-1}}{\rho}) + \boldsymbol{c}, \\
\boldsymbol{z}^j \leftarrow \boldsymbol{z}^{j-1} - \gamma\rho(\boldsymbol{y}^j - \boldsymbol{R}\boldsymbol{x}^j), \\
\boldsymbol{\nu}^j \leftarrow \boldsymbol{x}^j + \dfrac{\rho}{\mu}\boldsymbol{R}^\top(\boldsymbol{y}^j - \boldsymbol{R}\boldsymbol{x}^j - \dfrac{\boldsymbol{z}^j}{\rho}),
\end{cases}
$$

where γ is the update coffecient of Lagrangian multiplier

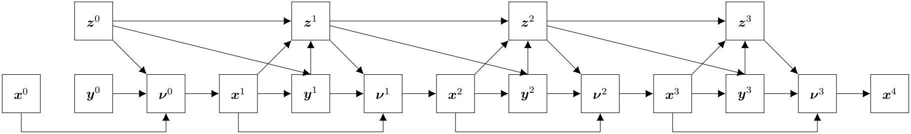


Figure 17: Data flow graph of ADMM, where $\boldsymbol{x}^0, \boldsymbol{y}^0, \boldsymbol{z}^0$ are fixed initial values, $\boldsymbol{s}$ is the input

ADMM-Net1

$$
\begin{cases}
\boldsymbol{x}_k^j \leftarrow \mathcal{C}_k(\boldsymbol{\nu}_k^{j-1}), k = 1, \ldots, K, \\
\boldsymbol{y}^j \leftarrow -\mathcal{P}_{\mathbb{R}_+^L}(\boldsymbol{R}\boldsymbol{x}^j - \sigma \odot \boldsymbol{z}^{j-1} + \boldsymbol{t}^j) + \boldsymbol{c}, \\
\boldsymbol{z}^j \leftarrow \boldsymbol{z}^{j-1} - \gamma(\boldsymbol{y}^j - \boldsymbol{R}\boldsymbol{x}^j) \oslash \sigma^j, \\
\boldsymbol{\nu}^j \leftarrow \boldsymbol{x}^j + \frac{\rho}{\mu}(\boldsymbol{W}^j)^\top(\boldsymbol{y}^j - \boldsymbol{R}\boldsymbol{x}^j - \sigma \odot \boldsymbol{z}^j),
\end{cases}
$$

where $\Theta = \{\boldsymbol{W}^j, \sigma^j, \boldsymbol{t}^j\}_{j=1}^T$ are the trainable weights

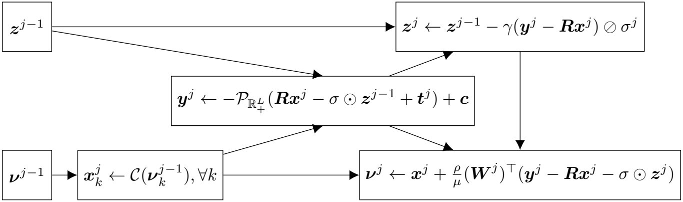


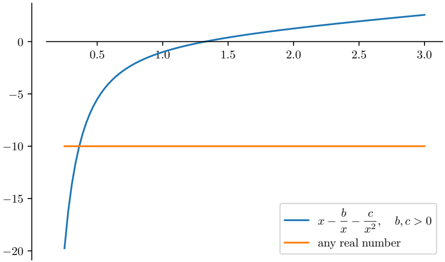
Figure 18: A typical layer of ADMM-Net.

ADMM-Net2

- Consider the general cubic equation

$$x^3 + ax^2 - bx - c = 0 \Leftrightarrow x - \frac{b}{x} - \frac{c}{x^2} = -a,$$

where $b, c > 0$



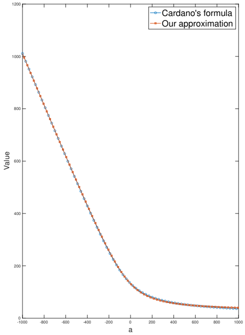
- Fixing b, c, denote the only positive root as $r(a)$. As illustrated, we have $r(a) \to 0$, as $a \to +\infty$, $r(a) \to -a$, as $a \to -\infty$

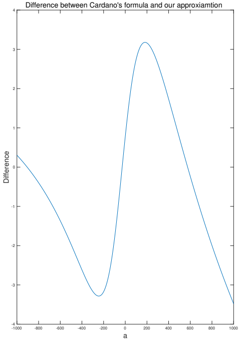
Approximation of $\mathcal{C}_k(\cdot)$

- We use a single branch of the rotated hyperbola to approximate it

$$[(a+m_k)+(y+n_k)](y+n_k)=\lambda_k$$

$$\Rightarrow \mathcal{A}_k(a;m_k,n_k,\lambda_k)=y=\sqrt{\frac{(a+m_k)^2}{4}+\lambda_k}-\frac{a+m_k}{2}-n_k$$





ADMM-Net2

$$
\begin{cases}
\boldsymbol{x}_k^j \leftarrow \mathcal{A}(\boldsymbol{\nu}_k^{j-1}; \lambda_k, m_k, n_k), k = 1, \ldots, K, \\
\boldsymbol{y}^j \leftarrow -\mathcal{P}_{\mathbb{R}_+^L}(\boldsymbol{R}\boldsymbol{x}^j - \sigma \odot \boldsymbol{z}^{j-1} + \boldsymbol{t}^j) + \boldsymbol{c}, \\
\boldsymbol{z}^j \leftarrow \boldsymbol{z}^{j-1} - \gamma(\boldsymbol{y}^j - \boldsymbol{R}\boldsymbol{x}^j) \oslash \sigma^j, \\
\boldsymbol{\nu}^j \leftarrow \boldsymbol{x}^j + \dfrac{\rho}{\mu}(\boldsymbol{W}^j)^\top(\boldsymbol{y}^j - \boldsymbol{R}\boldsymbol{x}^j - \sigma \odot \boldsymbol{z}^j),
\end{cases}
$$

where $\Theta = \{\boldsymbol{W}^j, \sigma^j, \boldsymbol{t}^j\}_{j=1}^T \cup \{\lambda_k, m_k, n_k\}_{k=1}^K$ are the trainable weights. $x^j(\boldsymbol{s}; \{\boldsymbol{W}^\tau, \sigma^\tau, \boldsymbol{t}^\tau\}_{\tau=1}^{j-1} \cup \{\lambda_k, m_k, n_k\}_{k=1}^K)$ is the output of ADMM-Net2 at j-th layer

Numerical Results

Table 2: classic ADMM vs deep unrolling ADMM in small example.

method	loss	obj	delay	fairness	load	iteration/layers
ADMM	0	-0.619	1.944	2.563	1.00	3207
ADMM-Net1	0.026	-2.389	0.298	2.687	31.45	1
ADMM-Net2	0.072	0.645	3.230	2.585	1.00	1
ADMM-Net1	0.022	-2.358	0.328	2.686	35.73	3
ADMM-Net2	0.074	0.589	3.179	2.590	1.05	3

Table 3: classic ADMM vs deep unrolling ADMM in large example.

method	loss	obj	delay	fairness	load	iteration/layers
ADMM	0	-183.355	4.377	187.732	1.00	20000
ADMM-Net1	0.152	-185.756	5.218	190.802	3.344	2
ADMM-Net2	0.249	-164.463	24.378	188.840	1.01	2
ADMM-Net1	0.128	-185.920	4.898	190.818	3.573	3
ADMM-Net2	0.248	-164.400	24.286	188.891	1.01	3

ADMM-Net2 as Warm-start

- ADMM-Net gives a fast approximate solution
- If we want to get a precise result, ADMM-Net becomes untrainable
- A natural idea is using ADMM-Net as warm-start