Learned Over-Parametrized Gradient Descent Method for Non-Negative Least Squares Problem

Akash Sen
Department of Mathematics
Indian Institute of Technology
Hyderabad, India
ma22resch11003@iith.ac.in

C. S. Sastry
Department of Mathematics
Indian Institute of Technology
Hyderabad, India
csastry@math.iith.ac.in

***Abstract*—The non-negative least squares (NNLS) problem finds a non-negative approximate solution to a linear system. Some well-studied iterative algorithms (such as projection gradient method) find the stated non-negative approximation. Notwithstanding this, a recent technique (known as algorithm unrolling) has emerged as a popular alternative, which maps an iterative algorithm into a deep unrolled network. In literature, the deep unrolled networks have attained importance due to their superiority in performance and interpretability as compared to their classical counterparts. In this work, we propose a framework based on the learned deep networks, (called learned over-parametrized gradient descent or LOGD, for brevity, method) for solving the NNLS problem in a data-driven setup. Numerically we show the promising results of the LOGD network and employ it in image denoising task. Our simulation results demonstrate that the proposed LOGD method performs better than the existing data-driven network and its classical (non-data driven) counterpart.**

***Index Terms*—Deep unrolling, Non-negative least squares, Inverse problem, Image denoising.**

I. Introduction

The non-negative least squares (NNLS) problem has relevance to numerous real-world and scientific applications including image processing [1], mining of text data [2], enhancement of speech data [3], spectral decomposition [4] [5], to name but a few. The NNLS problem deals with reconstructing a non-negative target vector that minimizes the discrepancies (in a norm sense) between the estimated values and the observed data. Mathematically the NNLS problem is stated as an optimization problem:

$$\min_{\mathbf{s}\in\mathbb{R}^n} \|F\mathbf{s}-\mathbf{x}\|_2^2 \;\text{ subject to }\; \mathbf{s}\geq 0, \tag{1}$$

where $F \in \mathbb{R}^{m\times n}$ is the forward transform, $\mathbf{x} \in \mathbb{R}^m$ possesses observed measurements in vector form, and $\mathbf{s} \in \mathbb{R}^n$ is the target vector to be reconstructed. In general, NNLS lacks closed form solution, which is in general not unique due to the non-negativity constraint. However, a variety of iterative methods are proposed to solve (1). Some of the popular methods include the interior point method [6], the active set based method [7] and the projected gradient descent method (PGDM) [8] etc. In a large scale setting, nevertheless, execution of these approaches tends to slow down primarily due to the need of inverting linear systems at each step. In PGDM, picking an optimal step-size is important in practice.

Deep unrolling [9] (or learned deep networks) is a recently developed strategy that has attracted attention of researchers due to its approximation potential over its known iterative counterparts. Neural networks are in general black-box in nature, whereas learned networks [10] are obtained from the iterative algorithms, resulting in an interpretable structure. In the literature, the potential of the learned networks has been realized in several applications such as sparse signal recovery [11], dictionary learning [12], image/signal denoising [13], etc.

The authors of [14] have developed an over-parametrized iterative solver for the NNLS problem. Recently, the authors of [10] have proposed an unrolled proximal gradient descent method (UPGDM) for this problem in a data-driven setup. Now driven by the potential of learned deep networks, we present a learned version of the algorithm proposed in [14]. We discuss its approximation capabilities and show that the method outperforms the UPGDM and its classical counterpart in the denoising application.

We use the following notations throughout the paper. For a positive integer n, the n-dimensional real space is denoted by $\mathbb{R}^n$. We use the lower-case letters and bold lower-case letters respectively for scalars and vectors, and capital letters for matrices. The ℓ_2 and ℓ_1 norms of a vector $\mathbf{x}$ are denoted respectively by $\|\mathbf{x}\|_2$ and $\|\mathbf{x}\|_1$. The number of non-zero components of a vector x is denoted by the symbol $\|.\|_0$. The symbol "o" is utilized as the composition between two functions. A vector whose all entries are zero is denoted as $\mathbf{0}$. The Hadamard product between two vectors $\mathbf{u} = (u_1,\ldots,u_n)^\top$ and $\mathbf{v} = (v_1,\ldots,v_n)^\top$ is defined as $\mathbf{u}\odot\mathbf{v} = (u_1\cdot v_1,\ldots,u_n\cdot v_n)^\top$.

We organize the paper as follows. In section II, we revisit the recent iterative algorithms that are applicable to the NNLS problem. While providing a brief account of learned algorithms, we present our learned over-parametrized gradient descent (LOGD) method in section III. In the last two sections, we provide the simulation results and concluding remarks.

II. NNLS through iterative solvers

In this section we present two iterative solvers for solving the NNLS problem.

The first author gratefully acknowledges the PMRF (Id: 2003328) being received by him.

A. NNLS via over-parametrization

In this subsection we give a brief overview of the over-parametrized iterative solver that can solve the NNLS problem. The authors of [14] have solved the NNLS problem by executing the gradient descent (GD) on the over-parametrized loss function

$$\mathcal{L}_{over}(\mathbf{s}) = \frac{1}{2}|| \ F\mathbf{s}^{\odot P} - \mathbf{x} \ ||_2^2 \tag{2}$$

with certain theoretical guarantees. The k^{th} GD update of the afore-mentioned loss function is

$$\mathbf{s}_{k+1} = \mathbf{s}_k - \eta_k \left[F^\top \left(F\mathbf{s}_k^{\odot P} - \mathbf{x}\right)\right] \odot \mathbf{s}_k^{\odot P-1} \tag{3}$$

with identical initialization and $P \geq 2$, for the learning rate η_k. Here, $s^{\odot P}$ stands for the Hadamard product of s taken P times.

B. NNLS via Proximal gradient descent method (PGDM)

In this method, the gradient descent iterates of the objective function in (1) are mapped onto the non-negative orthant to ensure the non-negativity of the approximate signal. The k^{th} iterate of PGDM is given by

$$\mathbf{s}_{k+1} = ReLU\big[(I - tF^\top F)\mathbf{s}_k + (tF^\top)\mathbf{x}\big], \tag{4}$$

where $t > 0$ is the step size and $ReLU$ is defined as

$$ReLU(\mathbf{z}) = \Big(\max\{z_1, 0\}, \ldots, \max\{z_n, 0\}\Big).$$

III. Learned over-parametrized gradient descent (LOGD) method

In this section, we provide a brief overview of learned algorithms and our proposed learned network that solves the NNLS problem.

A. Learned Algorithms

The concept of learned algorithms (or algorithm unrolling) has been introduced in [15]. The authors of this work have presented the learned iterative soft thresholding algorithm (LISTA), the unrolled version of the iterative hard thresholding algorithm (ISTA). LISTA has been shown to outperform ISTA. The idea behind the learned algorithms is to map each step of an iterative process of an algorithm into a network layer. Stacking a finite number of such layers gives rise to a learned (or unrolled) network. This enables the network to learn from the given data via backpropagation. One key advantage of unrolled networks lies in their interpretability. Unlike conventional deep learning architectures, which often function as "black boxes," learned networks retain a direct correspondence to iterative steps of the optimization. Additionally, learned architectures are typically data efficient.

B. Unrolled proximal gradient descent method (UPGDM)

The unrolled network [10], termed UPGDM, considers the l^{th} layer of the network as

$$\mathbf{s}_{k+1} = ReLU(\mathbf{s}_k - \mu_k W_1^k \mathbf{s}_k + \mu_k W_2^k \mathbf{x}), \tag{5}$$

where $W_1^k \equiv H^T H$, $W_2^k \equiv H^T$, $\mu_k \equiv t$ are the trainable parameters of the network. The l^{th} layer of the UPGDM network is shown in Algorithm 1. The overall architecture possessing several concatenated layers is shown in Algorithm 2.

Algorithm 1 UPGDM layer inference:

1: **Input:** $\mathbf{x} \in \mathbb{R}^m$, $\mathbf{s}_k \in \mathbb{R}^n$
2: $\mathbf{s}_{k+1} = ReLU\big(\mathbf{s}_k - \mu_k W_1^k \mathbf{s}_k + \mu_k W_2^k \mathbf{x}\big)$
3: **Output:** $\mathbf{s}_{k+1}$

Algorithm 2 UPGDM network inference:

1: **Input:** $\mathbf{x} \in \mathbb{R}^n$
2: **Initialize:** $\mathbf{s}_0 = \mathbf{0}$ and residual $\mathbf{r}_0 = \mathbf{s}$
3: **for** $i = 0, 1, 2, \ldots L-1$ **do**
4: $\quad \mathbf{s}_{i+1} = \texttt{UPGDMlayer}(\mathbf{x}, \mathbf{s}_i)$
5: **end for**
6: **Output:** Estimated non-negative signal $\hat{\mathbf{s}} = \mathbf{s}_L$

C. Proposed LOGD

As previously mentioned, our study aims to tackle the NNLS problem, stated in (1), in the unrolling setup. We convert each iteration step into a neural network layer by considering $W_1^l \equiv H^\top H$, $W_2^l \equiv H^\top$. The l^{th} layer of the network is given by

$$\mathbf{s}_{l+1} = \mathbf{s}_l - \eta_l \left[W_1^l \mathbf{s}_l^{\odot P} - W_2^l \mathbf{x}\right] \odot \mathbf{s}_l^{\odot P-1}.$$

Staking $L(> 2)$ layers together forms a deep LOGD network. It may be noted that the unrolled network layer does not involve any activation function. The non-linearlity, however, is introduced by the multiple application of the Hadamard product. For input $\mathbf{x} \in \mathbb{R}^m$, mathematically the LOGD model can be defined as

$$\begin{aligned}\mathcal{O}_{m,L,n} = \Bigg\{ & \mathcal{F}(\mathbf{x}; \mathbf{w}) = \mathbf{s}_L - \eta_L \left[W_1^L \mathbf{s}_L^{\odot P} - W_2^L \mathbf{x}\right] \odot \mathbf{s}_L^{\odot P-1}, \\ & \mathbf{s}_{l+1} = \mathbf{s}_l - \eta_l \left[W_1^l \mathbf{s}_l^{\odot P} - W_2^l \mathbf{x}\right] \odot \mathbf{s}_l^{\odot P-1}, \\ & W_1^l \in \mathbb{R}^{n \times n}, \mathbf{x} \in \mathbb{R}^m, \mathbf{s}_l \in \mathbb{R}^n, W_2^l \in \mathbb{R}^{n \times m}, \forall \, l \in [L], \\ & \mathbf{w} = vec([W_1^1 \ldots W_1^L][W_2^1 \ldots W_2^L]) \in \mathbb{R}^{n^2+mn} \Bigg\},\end{aligned} \tag{6}$$

where $\mathbf{s}_0$ is the initialized vector, and $\mathbf{w}$ is the set of trainable parameters in vector form. If $L > 2$, then any $\mathcal{F} \in \mathcal{O}_{m,L,n}$ is known as a deep LOGD network. Given the data set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{s}_i)\}_{i=1}^M$, our task is to find $\mathbf{w}^*$ such that $\mathcal{F}(\mathbf{x}_i; \mathbf{w}^*) \approx \mathbf{s}_i \ \forall \ i \in \{1, 2, \ldots, M\}$. The weight matrices W_1^l, W_2^l, and η_l are learned from the data. We provide the l^{th} layer of the network in Algorithm 3 and, by stacking the

layers, L (> 2) times, we form the deep LOGD network. Algorithm 4 summarizes the LOGD network.

Algorithm 3 l^{th} LOGD Network Layer:

Input: $\mathbf{x} \in \mathbb{R}^m$, $\mathbf{s}_l \in \mathbb{R}^n$
$\mathbf{s}_{l+1} = \mathbf{s}_l - \eta_l \left[W_1^l \mathbf{s}_l^{\odot P} - W_2^l \mathbf{x}\right] \odot \mathbf{s}_l^{\odot P-1}$
Output: $\mathbf{s}_{l+1}$

Algorithm 4 LOGD network inference:

Input: $\mathbf{x} \in \mathbb{R}^n$
Initialize: $\mathbf{s}_0 = \mathbf{0}$ and residual $\mathbf{r}_0 = \mathbf{s}$
for $i = 0, 1, 2, \ldots L-1$ **do**
    $\mathbf{s}_{i+1} = \texttt{UOGDlayer}(\mathbf{x}, \mathbf{s}_i)$
end for
Output: Estimated non-negative signal $\hat{\mathbf{s}} = \mathbf{s}_L$

IV. Simulation Results

In this section we describe our experimental setup and present an empirical demonstration of our LOGD network.

A. Training data

We have considered a random Gaussian matrix, F, of size 400×1200 with mean and variance being 0 and 1, respectively. We have generated 1000 samples of non-negative vectors with 0 mean and 1 variance. Then we have created the training data $\{\mathbf{x}_i, \mathbf{s}_i\}_{i=1}^{1000}$ by considering $\mathbf{x}_i = F\mathbf{s}_i$.

B. Training of LOGD network

For the training of $\mathcal{F}(\cdot, W)$, we have used the Adam optimizer [16] to minimize the mean squared error (MSE) loss function

$$L(W) = \frac{1}{1000} \sum_{i=1}^{1000} \|\mathcal{F}(\mathbf{x}_i, W) - \mathbf{s}_i\|_2^2. \tag{7}$$

The number of epochs that we have taken is 100000.

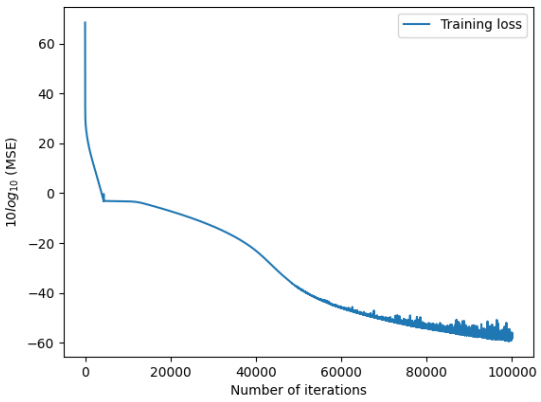


Fig. 1: Training Loss vs Epochs for LOGD.

C. Training loss against the number of Epochs

We now present behavior of the network's training loss in terms of the number of epochs. Figure 1 illustrates that, as the number of epochs increases, the training loss decreases progressively and approaches zero asymptotically. This confirms the successful training of the LOGD network. Further, Figure 2 and its zoomed version in Figure 3 show the prediction of the input signal and the ground-truth signal. These figures imply the successful prediction of the target signal by the LOGD.

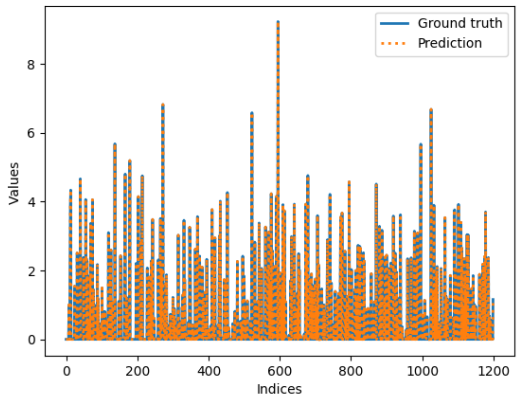


Fig. 2: Ground truth signal vs Predicted signal.

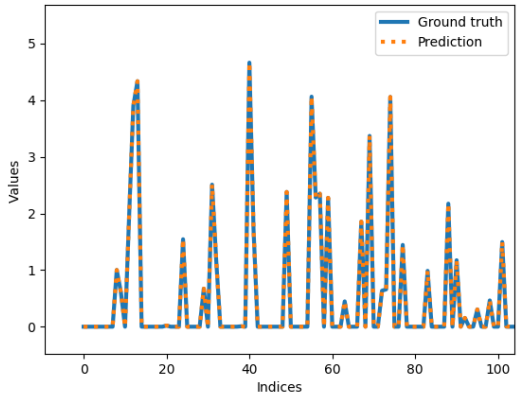


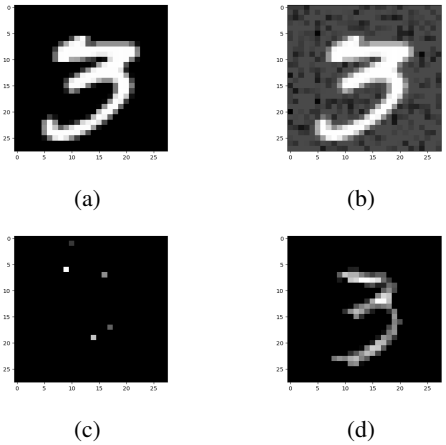
Fig. 3: Ground truth signal vs Predicted signal.

V. Application

In this section, we employ our LOGD model in a denoising application and compare it with the existing UPGDM model and its classical counterpart. We consider the denoising application on the MNIST data in the following setup. For our simulation work, we have collected 300 MNIST images consisting of '3' and denoted as $S_i^+ = \{\texttt{MNIST(3)}\}_i$. Then we have perturbed them as $\tilde{S}_i = S_i^+ - n^-$, where n^- is the positive white Gaussian noise. We have vectorized the

perturbed images, which are denoted as $\mathbf{s}_i = vec(\tilde{S}_i)$, for $i \in \{1, \ldots, 300\}$. In our simulations, we have considered the forward operator, F to be a random Gaussian matrix, leading to the training data $\{\mathbf{x}_i, \mathbf{s}_i^+\}_{i=1}^{300}$, where $\mathbf{x}_i = H\mathbf{s}_i$ and $\mathbf{s}_i^+ = vec(S_i^+)$.

The results in Figure 4 show the performances of the UPGD and LOGD methods, while the ones in Figure 5 also provide the results obtained via the over-parametrized gradient descent (OGD) and LOGD methods. From these results, it can be concluded that the LOGD has potential to become a viable method for image de-noising. It may be observed that a comparison in Figure 4 is provided between two data-driven methods, while in Figure 5 it is between a data-driven strategy and a non-data driven strategy.

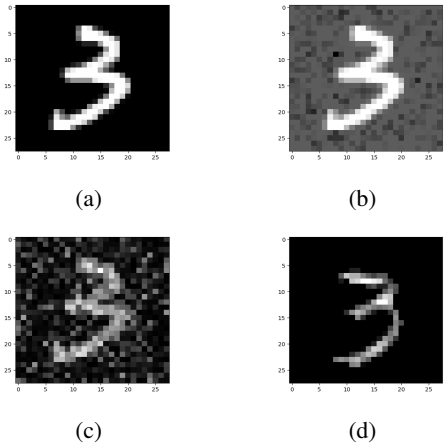


(a) (b) (c) (d)

Fig. 4: Comparison of the recovery of the original MNIST image "3" using the learned methods from the noisy image. Where (a) is the original ground truth image, (b) is the noisey image, and (c) and (b) are the recovered images from the UPGD and LOGD methods, respectively.

VI. Conclusions

In this work, we proposed a learned deep network model, named LOGD model, to address the NNLS problem in a data-driven setup. Further we employed our model in a denoising application and showed its superiority over its classical counterpart and an existing unrolled model, known as UPGDM. Our future effort will establish the convergence guarantees of this solver along with relevance to other inverse problems.



(a) (b) (c) (d)

Fig. 5: Comparison of the recovery of the original MNIST image "3" using the proposed learned method and its iterative counterpart, from the noisy image. Where (a) is the original ground truth image, (b) is the noisey image, and (c) and (b) are the recovered images from the OGD and LOGD methods, respectively.

References

[1] V. Monga and M. K. Mihçak, "Robust and secure image hashing via non-negative matrix factorizations." *IEEE Trans. Inf. Forensics Secur.*, vol. 2, no. 3-1, pp. 376–390, 2007.

[2] V. P. Pauca, F. Shahnaz, M. W. Berry, and R. J. Plemmons, "Text mining using non-negative matrix factorizations," in *Proceedings of the 2004 SIAM international conference on data mining*. SIAM, 2004, pp. 452–456.

[3] P. C. Loizou, "Speech enhancement based on perceptually motivated bayesian estimators of the magnitude spectrum," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5, pp. 857–869, 2005.

[4] C. Févotte, N. Bertin, and J.-L. Durrieu, "Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis," *Neural computation*, vol. 21, no. 3, pp. 793–830, 2009.

[5] P. Sajda, S. Du, T. R. Brown, R. Stoyanova, D. C. Shungu, X. Mao, and L. C. Parra, "Nonnegative matrix factorization for rapid recovery of constituent spectra in magnetic resonance chemical shift imaging of the brain," *IEEE transactions on medical imaging*, vol. 23, no. 12, pp. 1453–1465, 2004.

[6] S. Bellavia, M. Macconi, and B. Morini, "An interior point newton-like method for non-negative least-squares problems with degenerate solution," *Numerical Linear Algebra with Applications*, vol. 13, no. 10, pp. 825–846, 2006.

[7] C. L. Lawson and R. Hanson, "Linear least squares with linear inequality constraints," *Solving least squares problems*, pp. 158–173, 1974.

[8] R. A. Polyak, "Projected gradient method for non-negative least square," *Contemp Math*, vol. 636, pp. 167–179, 2015.

[9] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Processing Magazine*, vol. 38, no. 2, pp. 18–44, 2021.

[10] A. Sen, P. Pradhan, R. Randhi, and C. S. Sastry, "Unrolled proximal gradient descent method for non-negative least squares problem," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 7625–7629.

[11] S. Wu, A. Dimakis, S. Sanghavi, F. Yu, D. Holtmann-Rice, D. Storcheus, A. Rostamizadeh, and S. Kumar, "Learning a compressed sensing measurement matrix via gradient unrolling," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6828–6839.

[12] B. Tolooshams, A. Song, S. Temereanca, and D. Ba, "Convolutional dictionary learning based auto-encoders for natural exponential-family distributions," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9493–9503.

[13] Y. Li, M. Tofighi, J. Geng, V. Monga, and Y. C. Eldar, "Efficient and interpretable deep blind image deblurring via algorithm unrolling," *IEEE Transactions on Computational Imaging*, vol. 6, pp. 666–681, 2020.

[14] H.-H. Chou, J. Maly, and C. M. Verdun, "Non-negative least squares via overparametrization," *arXiv preprint arXiv:2207.08437*, 2022.

[15] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10. Madison, WI, USA: Omnipress, 2010, p. 399–406.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.