Learn the Approximation Distribution of Sparse Coding with Mixture Sparsity Network

Li Li[1], Xiao Long[2,3], Liansheng Zhuang[1,2(✉)], and Shafei Wang[4]

[1] School of Data Science, USTC, Hefei, China
lszhuang@ustc.edu.cn
[2] School of Information Science and Technology, USTC, Hefei, China
[3] Peng Cheng Laboratory, Shenzhen, China
[4] Northern Institute of Electronic Equipment, Beijing, China

Abstract. Sparse coding is typically solved by iterative optimization techniques, such as the ISTA algorithm. To accelerate the estimation, neural networks are proposed to produce the best possible approximation of the sparse codes by unfolding and learning weights of ISTA. However, due to the uncertainty in the neural network, one can only obtain a possible approximation with fixed computation cost and tolerable error. Moreover, since the problem of sparse coding is an inverse problem, the optimal possible approximation is often not unique. Inspired by these insights, we propose a novel framework called Learned ISTA with Mixture Sparsity Network (LISTA-MSN) for sparse coding, which learns to predict the best possible approximation distribution conditioned on the input data. By sampling from the predicted distribution, LISTA-MSN can obtain a more precise approximation of sparse codes. Experiments on synthetic data and real image data demonstrate the effectiveness of the proposed method.

Keywords: Sparse coding · Learned ISTA · Mixture Sparsity Network

1 Introduction

Sparse coding (SC) has shown great success in uncovering global information from noisy and high dimensional data such as image super-resolution [17], image classification [15], and object recognition [13]. The main goal of SC is to find the sparse representation from over-complete dictionary. To solve this problem, classic methods are based on high dimensional optimization theory, such as proximal coordinate descent [9], Least Angle Regression [8] and proximal splitting methods [2]. Among these methods, Iterative Shrinkage-Thresholding Algorithm (ISTA) [5] is the most popular one, which belongs to proximal-gradient method. It has been proven that ISTA converges with rate $1/t$, where t is the number of the iterations/layers, and its computational complexity is too high. To address this issue,

L. Li and X. Long—Contributed equally.

H. Ma et al. (Eds.): PRCV 2021, LNCS 13022, pp. 387–398, 2021.
https://doi.org/10.1007/978-3-030-88013-2_32

Beck and Teboulle proposed the FISTA [3] algorithm converges more rapidly in 2009. The major difference with ISTA is the introduction of a "momentum" term. The algorithms mentioned above belong to traditional iterative approaches. However, even though the FISTA accelerates the speed of convergence a lot, these algorithms are still too slow for practical applications such as real-time object recognition.

To further increase the speed of ISTA, Gregor and LeCun proposed a trainable version of ISTA called Learned ISTA (LISTA) [11] by unfolding ISTA structure into a recurrent neural network (RNN). Unlike the traditional iterative approaches, LISTA is highly computationally efficient during the inference period. Once the learnable parameters of the neural network are trained, it can quickly estimate a solution of sparse codes by passing the input through a fixed recurrent neural network rather than solving a series of convex optimization problems. Moreover, LISTA yields better estimation result than ISTA on new samples for the same number of iterations/layers. This idea has led to a profusion of literature [6,18]. For one thing, they follow the idea of LISTA and modify the structure to use more historical information. For another, some works change shared weights to layer-wise weights and get better performance. All these methods have achieved impressive performance in solving sparse coding.

But the existing neural network-based methods have suffered from the following drawbacks. First, they ignore the fact that the LASSO problem is an inverse problem and the optimal solution may be not unique. For example, in a super resolution task [12] a low-resolution image could be explained by many different high-resolution images. In this case, the neural network designed to predict a specific value is not effective. Second, due to the uncertainty of deep learning, it is hard to get a unique accurate solution of sparse codes with fixed iterations. In fact, neural network-based methods (e.g., LISTA) can only obtain a possible approximation of sparse codes with fixed computational cost and tolerable error [11]. Therefore, the best possible approximation of sparse codes should satisfy some kind of distribution. We argue that, it is more reasonable to predict the distributions of possible approximation of sparse codes instead of the specific value of best possible approximation.

Inspired by the above insights, this paper proposes a novel framework called Learned ISTA with Mixture Sparsity Network (LISTA-MSN) for sparse coding. The key idea is to introduce a novel module called Mixture Sparsity Network (MSN) to predict the distribution of best possible approximation of sparse codes. Specifically, the proposed framework first uses the popular LISTA network to obtain an initial estimate solution of sparse coding. The LISTA framework can be replaced by any other neural networks such as ALISTA [14], Coupled-LISTA (LISTA-CP) [6], etc. Then, our framework introduces the Mixture Sparsity Network to predict the distribution of the best possible approximations according to the initial estimate solution. At last, the proposed framework samples the final optimal results of sparse coding from the distribution. Note that, it is a challenging problem to model the conditional probability density function of the target sparse vectors conditioned on the input vector. To address this problem,

Mixture Sparsity Network adds an additional penalty to the popular framework of Mixture Density Network [4] so that it can ensure the sparsity of network's outputs as we hope. Experiments on synthetic data and real image data have shown that the proposed LISTA-MSN framework can significantly improve the accuracy and convergence speed of sparse coding.

In summary, our main contributions are as follows:

- A novel framework called LISTA-MSN is proposed to learn fast approximations of sparse codes. Different from previous works (such as LISTA [11], SC2Net [18] and ALISTA [14]), the proposed framework learns to predict the distribution of the best possible approximation of sparse codes instead of the best possible approximation. To the best of our knowledge, this is the first work to learn a trainable neural network to predict the distribution of the optimal solution of sparse coding. Moreover, the proposed framework is very flexible, where the LISTA can be replaced with any other neural network for sparse coding.
- Mixture Sparsity Network is proposed to model the conditional probability density function of the target sparse vectors conditioned on the input vector. It ensures that the data sampling from the predicted distribution is sparse enough, which makes the final results meaningful.

2 The Proposed Method

In this section, we will introduce the LISTA-MSN framework. The architecture of LISTA-MSN is shown in Fig. 1. Specifically, it first uses the LISTA network to obtain a coarse estimation of sparse codes. Then, the framework introduces the mixture sparsity network to predict the distribution of the best possible approximations according to the initial estimation. Finally, the framework samples the final optimal sparse codes from the distribution. Note that, the proposed method adds penalty to the mixture density network [4] to ensure the sparsity of results.

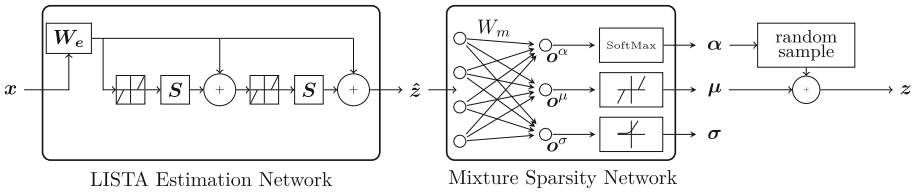

Fig. 1. The architecture of LISTA-MSN.

2.1 LISTA Network

Given the input data $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N] \in \mathbb{R}^{n \times N}$, sparse coding aims to learn the over-complete dictionary $\boldsymbol{D} = [\boldsymbol{d}_1, \boldsymbol{d}_2, ..., \boldsymbol{d}_m] \in \mathbb{R}^{n \times m}$ and the sparse representation $\boldsymbol{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_N] \in \mathbb{R}^{m \times N}$. And the goal of LISTA Network is to

calculate the coarse estimation $\hat{\boldsymbol{Z}} = [\hat{\boldsymbol{z}}_1, \hat{\boldsymbol{z}}_2, ..., \hat{\boldsymbol{z}}_N] \in \mathbb{R}^{m \times N}$, where $\hat{\boldsymbol{z}}_q$ is the coarse estimation of sparse codes of $\boldsymbol{x}_q$ with limited iterations. Specifically, LISTA is:

$$\boldsymbol{z}^{(t+1)} = h_{\boldsymbol{\theta}}(\boldsymbol{W_e}\boldsymbol{x} + \boldsymbol{S}\boldsymbol{z}^{(t)}) \quad t = 0, ..., K-1 \tag{1}$$

$\hat{\boldsymbol{z}} = \boldsymbol{z}^{(K)}$, and K is fixed number of steps. $\boldsymbol{\theta} = [\theta_1, ..., \theta_m]$ is a trainable vector. The variables $\boldsymbol{W_e}$, $\boldsymbol{S}$ and $\boldsymbol{\theta}$ are learned by the given training data. So, the LISTA can be replaced by any other neural networks such as ALISTA [14], Coupled-LISTA (LISTA-CP) [6], etc.

2.2 Mixture Sparsity Network

Having the coarse estimation $\hat{\boldsymbol{z}}$ generated from LISTA Network, in this stage, the mixture sparsity network is proposed to model the conditional distribution of sparse codes $\boldsymbol{z}$ on the coarse estimation $\hat{\boldsymbol{z}}$ as the linear combination of kernel function:

$$P(\boldsymbol{z}|\hat{\boldsymbol{z}}) = \sum_{i=1}^{M} \alpha_i(\hat{\boldsymbol{z}})\phi_i(\boldsymbol{z}|\hat{\boldsymbol{z}}) \tag{2}$$

where M is the number of kernel function, ϕ_i is the kernel function, $\alpha_i(\hat{\boldsymbol{z}})$ is the mixture coefficients, which can be regarded as prior probability of the sparse code $\boldsymbol{z}$ being generated from i^{th} kernel given the coarse estimation $\hat{z}$. In practice, the Gauss density function is often chosen as the kernel function:

$$\phi_i(\boldsymbol{z}|\hat{\boldsymbol{z}}) = \frac{\exp}{(2\pi)^{\frac{m}{2}} \prod_{j=1}^{m} \sigma_{ij}(\hat{\boldsymbol{z}})} \left\{ -\sum_{j=1}^{m} \frac{(z_j - \mu_{ij}(\hat{\boldsymbol{z}}))^2}{2\sigma_{ij}(\hat{\boldsymbol{z}})^2} \right\} \tag{3}$$

where z_j is the j^{th} element of target data $\boldsymbol{z}$, μ_{ij} and σ_{ij} denote the j^{th} element of mean and standard deviation of the i^{th} kernel. A diagonal matrix is used instead of the covariance matrix of Gauss density function from the concern of computational cost. The parameters $\Theta = \{\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\sigma}\}$ are outputs of network which depend on the coarse estimation $\hat{\boldsymbol{z}}$. Since the $\hat{\boldsymbol{z}}$ is calculated by LISTA Network, the distribution of the sparse code is conditioned on input data. A simple full-connected network is constructed to generate the parameters $\Theta = \{\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\sigma}\}$ like Fig. 1. Different activation functions are applied to these parameters in order to satisfy the restrictions. For the coefficient $\boldsymbol{\alpha}(\hat{\boldsymbol{z}})$, $\alpha_i > 0$ and $\sum_{i=1}^{M} \alpha_i = 1$, we use the SoftMax function [10]. For the standard deviation of kernel $\boldsymbol{\sigma}(\hat{\boldsymbol{z}})$ which should be positive, we choose a modified Elu function [7]:

$$f(t) = \begin{cases} t + 1, & \text{if } t \geq 0; \\ \gamma[exp(t) - 1] + 1, & \text{otherwise} \end{cases} \tag{4}$$

where γ is a scale parameter.

Next, to ensure the sparsity of the final result, it is noteworthy that the element of $\boldsymbol{\mu}$ which corresponds to zero elements in sparse codes should be very

close to zero when the network was trained. So, we use the coordinate-wise shrinking function h_ϵ as sparsity restriction of $\boldsymbol{\mu}$ to keep it sparse:

$$h_\epsilon(t) = sign(t)(|t| - \epsilon)_+ \tag{5}$$

Here, ϵ is a hyperparameter with the positive value which could be set to a small number *e.g.*, 0.01. The shrinking function h_ϵ could change the small number to zero. If the output of linear layer id denoted as $\boldsymbol{o} \in \mathbb{R}^{(2m+1)M} = \{o_i^\alpha, o_{i1}^\mu, ..., o_{im}^\mu, o_{i1}^\sigma, ..., o_{im}^\sigma\}_{i=1}^M$, the parameters $\boldsymbol{\Theta}$ are:

$$\alpha_i = \frac{exp(o_i^\alpha)}{\sum_{j=1}^M exp(o_j^\alpha)} \quad \mu_{ij} = h_\epsilon(o_{ij}^\mu) \quad \sigma_{ij} = Elu(o_{ij}^\sigma) \tag{6}$$

The whole network is optimized by using the negative log-likelihood loss (NLL) of (4) and (5), the loss function is defined as:

$$\begin{aligned} E &= -\sum_{q=1}^N E^q \\ E^q &= -ln(\sum_{i=1}^M \alpha_i(\hat{\boldsymbol{z}}_q)\phi_i(\boldsymbol{z}_q|\hat{\boldsymbol{z}}_q)) \end{aligned} \tag{7}$$

So, the derivatives of E^q with respect to output of linear layer $\boldsymbol{o}$ are calculated as follows:

$$\begin{aligned} \pi_i &= \frac{\alpha_i\phi_i}{\sum_{j=1}^M \alpha_j\phi_j} \\ \frac{\partial E^q}{\partial o_j^\alpha} &= \alpha_i - \pi_i \end{aligned} \tag{8}$$

$$\frac{\partial E^q}{\partial o_{ij}^\sigma} = \pi_i \left\{ \frac{1}{\sigma_{ij}} - \frac{(z_j - \mu_{ij})^2}{\sigma_{ij}^3} \right\} f'(o_{ij}^\sigma) \tag{9}$$

$$\frac{\partial E^q}{\partial o_{ij}^\mu} = \pi_i \left\{ \frac{(\mu_{ij} - z_j)}{\sigma_{ij}^2} \right\} \delta(|o_{ij}^\mu| > \lambda) \tag{10}$$

where f' denotes the derivative of function f, $\delta(t)$ is defined as follow:

$$\delta(t) = \begin{cases} 1, & \text{if } t \text{ is true;} \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

So, the standard back-propagation is guaranteed, and the algorithm is shown in Algorithm 1 and Algorithm 2.

Finally, after getting the parameters Θ, we need to sample the final results from the learned distribution. Luckily, we get the mixture of Guass density function by which the sampling approach can be easily implemented. The location is obtained by randomly sampling according to $\boldsymbol{\alpha}$ and the corresponding center is

the result of sampling. Actually, what interested us more is the most likely value of the output. The most likely value for output vector is calculated by maximum the conditional distribution, but this procedure might be computationally costly. A faster approximation is to use the center of kernel function which has highest probability of being sampled, from (4) and (5):

$$ind = \arg\max_i \{\alpha_i\} \tag{12}$$

The corresponding center μ_{ind} is the most likely output. And the result of sample will be sparse as long as the h_ϵ activation for $\boldsymbol{\mu}$ is used.

Algorithm 1. LISTA-MSN: fprop and bprop

fprop$(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{W_e}, \boldsymbol{S}, \boldsymbol{\theta})$
Variable $Z(t), C(t), B, \boldsymbol{\Theta}$ are stored for bprop
$B = \boldsymbol{W_e x}$; $Z(0) = h_{\boldsymbol{\theta}}(B)$
for t=1 to K **do**
$C(t) = B + \boldsymbol{S}Z(t-1)$
$Z(t) = h_{\boldsymbol{\theta}}(C(t))$
end for
$\hat{\boldsymbol{z}} = Z(K)$
$\boldsymbol{o} = \boldsymbol{W_m}\hat{\boldsymbol{z}}$
$\alpha_i = \frac{exp(\hat{o}_i^\alpha)}{\sum_{j=1}^{M} exp(\hat{o}_j^\alpha)}$ $\mu_{ij} = h_\epsilon(\hat{o}_{ij}^\mu)$ $\sigma_{ij} = f(\hat{o}_{ij}^\sigma)$
$\boldsymbol{\Theta} = \{\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\sigma}\}$

bprop $(\boldsymbol{z^*}, \boldsymbol{x}, \boldsymbol{W_e}, \boldsymbol{S}, \boldsymbol{W_m}, \boldsymbol{\theta}, \delta\boldsymbol{W_e}, \delta\boldsymbol{S}, \delta\boldsymbol{W_m}, \delta\boldsymbol{\theta})$
$Z(t)$, $C(t)$, B and $\boldsymbol{\Theta}$ were stored in fprop
Initialize: $\delta B = 0$; $\delta\boldsymbol{S} = 0$; $\delta\boldsymbol{\theta} = 0$
$\delta\boldsymbol{o}$ is calculated by (10) − (12)
$\delta\boldsymbol{W_m}(t) = \delta\boldsymbol{o}\hat{\boldsymbol{z}}^T$; $\delta\hat{\boldsymbol{z}} = \boldsymbol{W_m}^T\delta\boldsymbol{o}$; $\delta Z(K) = \delta\hat{\boldsymbol{z}}$
for $t = K$ down to 1 **do**
$\delta C(t) = h_\theta{}'(C(t)) \odot \delta Z(t))$
$\delta\theta = \delta\theta - sign(C(t)) \odot \delta C(t)$
$\delta B = \delta B + \delta C(t)$
$\delta\boldsymbol{S} = \delta\boldsymbol{S} + \delta C(t)Z(t-1)^T$
$\delta Z(t-1) = \boldsymbol{S}^T\delta C(t)$
end for
$\delta B = \delta B + h_\theta{}'(B) \odot \delta Z(0)$
$\delta\theta = \delta\theta - sign(B) \odot h_\theta(B)\delta Z(0)$
$\delta W_e = \delta B X^T$

3 Experiment

In this section, we evaluate the performance of the proposed framework on synthetic data and real image data, and compare it with other state-of-the-art algorithms, including ISTA, FISTA, LISTA, ALISTA, and Coupled-LISTA. Similar to previous works [1,3,11], we adopt Prediction error and Function value error as

the criteria. Prediction error is the squared error between the input data and predicted codes in a certain dictionary. And cost function error is the cost between current prediction and best prediction. These criteria can evaluate the accuracy and convergence of the algorithm. During the experiments, we find that all the evaluated methods perform stable when λ ranges between 0.1 and 0.9. So, the sparsity parameter $\lambda = 0.5$ is fixed in all the experiments, which means the difference during experiments is just the algorithms themselves. All the algorithms are implemented by python and pytorch, and run the experiments on a DGX-1v server.

3.1 Synthetic Experiments

For synthetic case, the dictionary $D \in \mathbb{R}^{n \times m}$ of standard normal distribution is generated. Once D is fixed, a set of Gaussian i.i.d. samples $\{x_i\}_{i=1}^{N} \in \mathbb{R}^n$ are draw. In this case, we set $m = 256$, $n = 64$, $N_{train} = 40000$ and $N_{test} = 10000$.

In this subsection, two experiments are carried out to verify the performance of the proposed method. First, we compare the LISTA-MSN with several traditional algorithms by measuring their prediction error to verify whether LISTA-MSN outperforms the traditional iterative approaches. Figure 2 shows the prediction error of several algorithms for different iterations or layers (For ISTA and FISTA, the number of iterations changes. For LISTA and LISTA-MSN, the number of layers changes.) From Fig. 2, we can observe that LISTA-MSN can always achieve the lowest prediction error in different depth compared with other algorithms. Even a 10 layers LISTA-MSN can achieve the same error as the five times depth of LISTA or FISTA. Furthermore, in order to verify whether the proposed framework is effective under different estimation network structures. So, we use the proposed framework for verification under the two network structures (ALISTA and Coupled-LISTA) respectively. Figure 3 and Fig. 4 show that for different estimation network structures, the proposed framework also outperforms the original network structure. And it can decrease the prediction error nearly by 10% at the same network depth and structure.

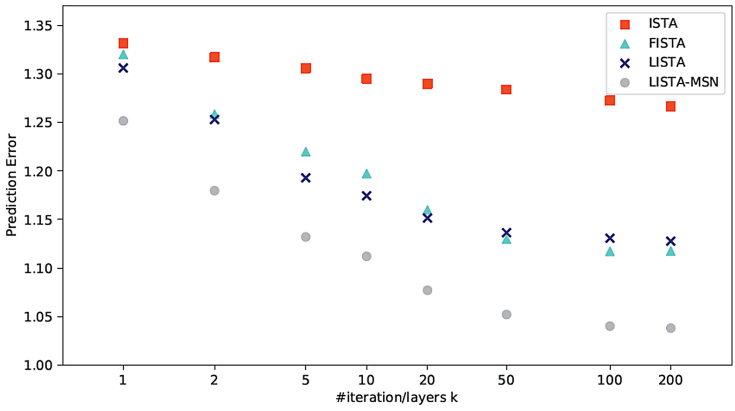

Fig. 2. Evolution of prediction error in ISTA, FISTA, LISTA, LISTA-MSN on simulated data

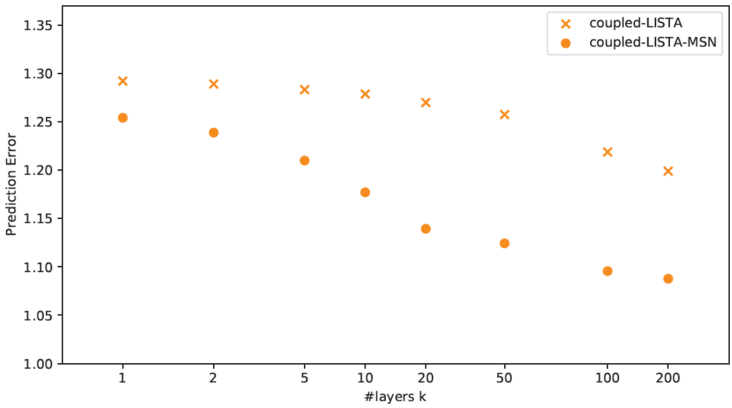

Fig. 3. Evolution of prediction error in Coupled-LISTA, Coupled-LISTA-MSN on simulated data

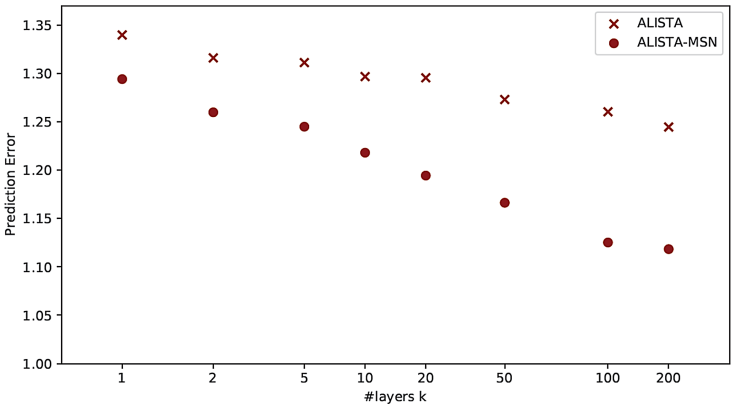

Fig. 4. Evolution of prediction error in ALISTA, ALISTA-MSN on simulated data

Second, we compare the convergence performance of the proposed method with other algorithms by calculating the function value error. Figure 5 and Fig. 6 show that the proposed method converges faster than traditional iterative approaches (ISTA and FISTA) and reaches a lower overall cost than other methods. Besides, we can also find that under the same network structure, using the proposed framework can improve the convergence speed of the original algorithm.

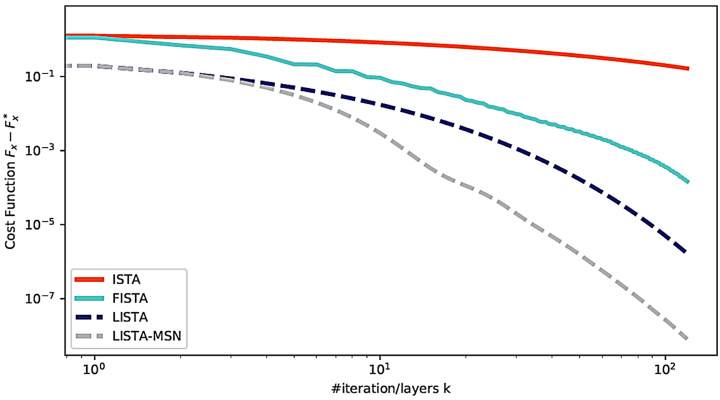

Fig. 5. Evolution of the function value error in ISTA, FISTA, LISTA, LISTA-MSN on simulated data

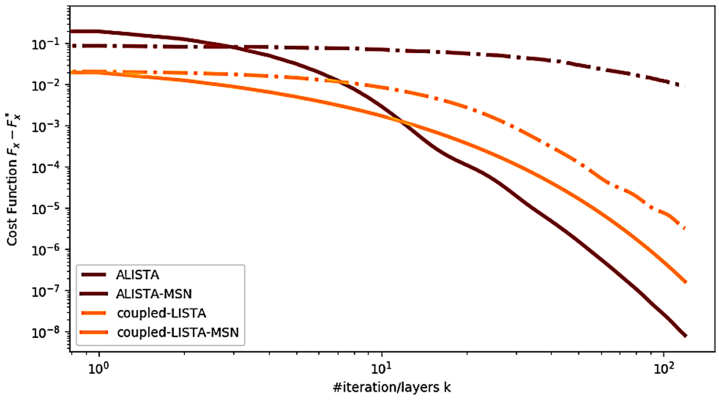

Fig. 6. Evolution of the function value error in four algorithms on simulated data

3.2 Digit Data Experiments

About real image data, we use the handwritten digits dataset from scikit-learn [Pedregosa et al., 2011]. The digits dataset contains 60,000 training images and 10,000 test images, where each image size is 8×8 and sampled from digits (0 to 9). We randomly sample $m = 256$ samples from dataset and normalize it to generate the dictionary D. Besides the above, all the image data is processed to remove its mean and normalize its variance.

In the digit dataset, we also compare several algorithms with the proposed method by measuring their prediction error. From Fig. 7, 8 and Fig. 9, we can observe that the results are very similar to the synthetic data. The proposed method outperforms the traditional iterative approaches. And under different estimation network structures (LISTA, ALISTA and Coupled-LISTA) with the use of our framework, the prediction error can also decrease. The interesting

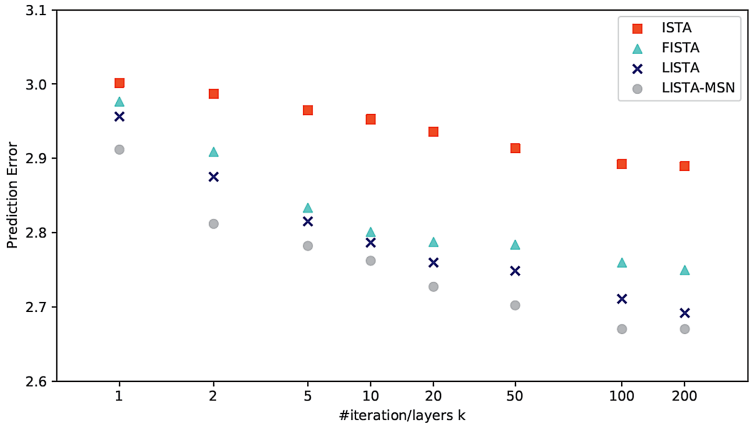

Fig. 7. Evolution of prediction error in ISTA, FISTA, LISTA, LISTA-MSN on digit data

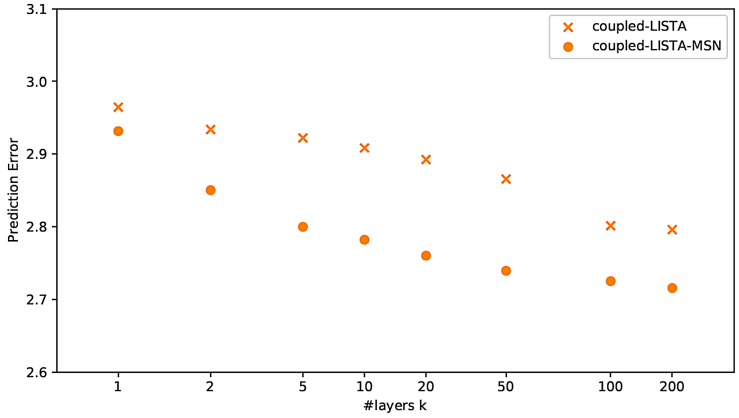

Fig. 8. Evolution of prediction error in Coupled-LISTA, Coupled-LISTA-MSN on digit data

thing observed is that all the algorithms perform better on synthetic dataset than digit dataset. For this result, the dictionary D of the digit data has a much richer correlation structure than the simulated Gaussian dictionary, which is known to impair the performance of learned algorithms [16].

3.3 Sparsity Analysis

As mentioned earlier, how to guarantee the sparsity of results from the learned distribution is a very important thing. And the sparse regulation is put forward to solve this problem. Therefore, we experimented to prove the importance of sparse regulation to MSN by comparing the sampling results' sparsity of the

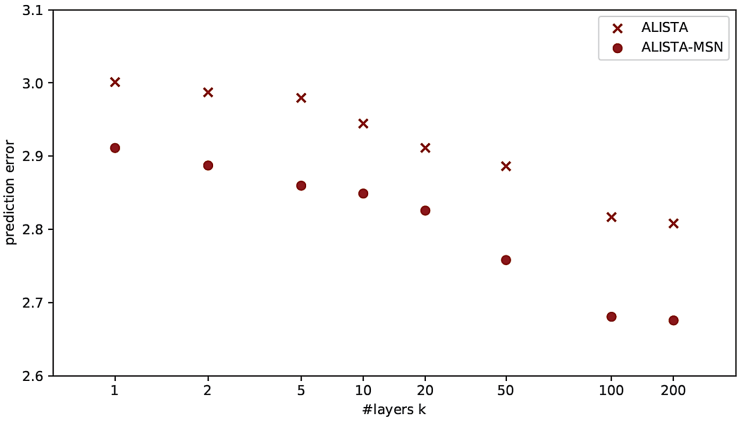

Fig. 9. Evolution of prediction error in ALISTA, ALISTA-MSN on digit data

proposed method with or without sparse regulation under the same estimation network structure. We calculate the proportion of zero elements in the final results. Table 1 shows that no matter how the dataset changes, under a certain estimation network structure with sparse regulation can always keep the sparsity of the results. This also guarantees that the final results are meaningful.

Table 1. The sparsity of the results whether use sparse regulation under a certain network structure

Datasets	Whether use sparse regulation	LISTA-MSN	ALISTA-MSN	Coupled-LISTA-MSN
Synthetic data	Without	35.67%	38.52%	40.28%
	With	**97.57%**	**94.31%**	**95.46%**
Digit data	Without	34.30%	33.60%	37.59%
	With	**94.67%**	**93.69%**	**93.55%**

4 Conclusion

This paper proposes a novel framework called LISTA-MSN for sparse coding, which significantly improves the accuracy of sparse coding. Different from existing neural network-based methods, the proposed framework learns the approximation distribution of sparse codes, then obtains the final optimal solution by sampling from the learned distribution. Furthermore, the proposed framework is very flexible. Mixture sparsity network can be combined with various neural networks as estimation networks for sparse coding, including LISTA, ALISTA, and Coupled-LISTA. Experimental results show that the proposed framework can significantly improve the accuracy of sparse coding.

Acknowledgment. This work was supported in part to Dr. Liansheng Zhuang by NSFC under contract (U20B2070, No. 61976199), and in part to Dr. Houqiang Li by NSFC under contract No. 61836011.

References

1. Ablin, P., Moreau, T., Massias, M., Gramfort, A.: Learning step sizes for unfolded sparse coding. In: Advances in Neural Information Processing Systems, pp. 13100–13110 (2019)
2. Bauschke, H.H., Combettes, P.L., et al.: Convex Analysis and Monotone Operator Theory in Hilbert Spaces, vol. 408. Springer, New York (2011). https://doi.org/10.1007/978-1-4419-9467-7
3. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Imaging Sci. **2**(1), 183–202 (2009)
4. Bishop, C.M.: Mixture density networks (1994)
5. Blumensath, T., Davies, M.E.: Iterative thresholding for sparse approximations. J. Fourier Anal. Appl. **14**(5–6), 629–654 (2008). https://doi.org/10.1007/s00041-008-9035-z
6. Chen, X., Liu, J., Wang, Z., Yin, W.: Theoretical linear convergence of unfolded ISTA and its practical weights and thresholds. In: Advances in Neural Information Processing Systems, pp. 9061–9071 (2018)
7. Clevert, D.A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (ELUs) (2015)
8. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al.: Least angle regression. Ann. Stat. **32**(2), 407–499 (2004)
9. Friedman, J., Hastie, T., Höfling, H., Tibshirani, R., et al.: Pathwise coordinate optimization. Ann. Appl. Stat. **1**(2), 302–332 (2007)
10. Gold, S., Rangarajan, A., et al.: Softmax to softassign: neural network algorithms for combinatorial optimization. J. Artif. Neural Netw. **2**(4), 381–399 (1996)
11. Gregor, K., LeCun, Y.: Learning fast approximations of sparse coding. In: Proceedings of the 27th International Conference on International Conference on Machine Learning, pp. 399–406 (2010)
12. Ledig, C., et al.: Photo-realistic single image super-resolution using a generative adversarial network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017
13. Lee, H., Ekanadham, C., Ng, A.Y.: Sparse deep belief net model for visual area V2. In: NIPS (2007)
14. Liu, J., Chen, X., Wang, Z., Yin, W.: ALISTA: analytic weights are as good as learned weights in LISTA. In: International Conference on Learning Representations (2019)
15. Mairal, J., Bach, F.R., Ponce, J., Sapiro, G., Zisserman, A.: Discriminative learned dictionaries for local image analysis. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2008)
16. Moreau, T., Bruna, J.: Understanding trainable sparse coding via matrix factorization. Stat **1050**, 29 (2017)
17. Yang, J., Wright, J., Huang, T.S., Ma, Y.: Image super-resolution via sparse representation. IEEE Trans. Image Process. **19**(11), 2861–2873 (2010). https://doi.org/10.1109/TIP.2010.2050625
18. Zhou, J.T., et al.: SC2Net: sparse LSTMs for sparse coding. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)