JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 29, Number 1, 2022

Pp. 1–11
DOI: 10.1089/cmb.2021.0438

RECOMB 2021

Deep Unfolding for Non-Negative Matrix Factorization with Application to Mutational Signature Analysis

RAMI NASSER,[1,i] YONINA C. ELDAR,[2] and RODED SHARAN[1]

ABSTRACT

Non-negative matrix factorization (NMF) is a fundamental matrix decomposition technique that is used primarily for dimensionality reduction and is increasing in popularity in the biological domain. Although finding a unique NMF is generally not possible, there are various iterative algorithms for NMF optimization that converge to locally optimal solutions. Such techniques can also serve as a starting point for deep learning methods that unroll the algorithmic iterations into layers of a deep network. In this study, we develop unfolded deep networks for NMF and several regularized variants in both a supervised and an unsupervised setting. We apply our method to various mutation data sets to reconstruct their underlying mutational signatures and their exposures. We demonstrate the increased accuracy of our approach over standard formulations in analyzing simulated and real mutation data.

Keywords: NMF, unfold and deep network.

1. INTRODUCTION

Non-negative matrix factorization (NMF) is a popular and useful decomposition tool for high-dimensional data. It is widely used in signal and image processing, text analysis, and in analyzing DNA mutation data. NMF is NP-hard (Vavasis, 2010) in general, and is commonly approximated by various iterative algorithms such as multiplicative updates (MUs) (Lee and Seung, 2000) and alternating non-negative least squares (Lin, 2007). Almost all NMF methods use a two-block coordinate descent scheme, which alternatively optimizes one of the W, H matrices in the data decomposition $V \sim WH$, whereas keeping the other fixed (Gillis, 2014). These iterative algorithms generally suffer from slow convergence and high computational cost when applied to large matrices (Kim and Park, 2011).

Recently, architectures based on deep learning were suggested for NMF (Hershey et al., 2014; Wisdom et al., 2017) as part of a general unfolding (or unrolling) framework (Monga et al., 2021). Unrolling techniques connect between iterative methods and deep networks by viewing each iteration of an underlying iterative algorithm as a layer of a network, such that concatenating the layers forms a deep neural network where the algorithm parameters transfer to the network parameters. The network is trained using backpropagation, resulting in model parameters that are learned from real-world training sets. However, these previous unrolling methods for NMF were limited to supervised settings where one of the matrix factors is known and can be used for training.

[1]Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv, Israel.
[2]Department of Math and Computer Science, Weizmann Institute of Science, Rehovot, Israel.
[i]ORCID ID (https://orcid.org/0000-0003-2670-6856).

In this study, we develop a deep unrolled network architecture, which we call deep non-negative matrix factorization (DNMF), for regularized variants of NMF for both the supervised and unsupervised settings. In our model, we learn two types of weight matrices for added flexibility in learning complex patterns and design the network so that conventional backpropagation tools such as the auto gradient in Pytorch can be used to allow for large-scale implementation. We implement the resulting networks and show their utility over standard iterative formulations. In particular, we apply our constructions to analyze a diverse collection of simulated and real mutation data sets, and show that they lead to better reconstructions of unseen data compared with the MU scheme. In the supervised setting, we train the network based on given input vectors V and their corresponding coefficients H, without the need of knowing the underlying dictionary (corresponding to mutational signatures) W. In the unsupervised setting, our network operates with the input non-negative data matrix V only.

2. METHODS

2.1. Problem formulation and current approaches

NMF receives as input a non-negative matrix $V_{f\times n}$ and a number k of desired factors; its goal is to decompose V into a product of two non-negative matrices $W_{f\times k}$ and $H_{k\times n}$ such that $\|V-WH\|_2$ is minimized. In addition to the popular Euclidean distance criterion, often called *reconstruction error*, another popular optimization criterion for NMF is the Kullback–Leibler (KL) divergence $KL(V\|WH)=\sum_{ij}(V_{ij}\log(\frac{V_{ij}}{(WH)_{ij}})-V_{ij}+(WH)_{ij})$. In the presentation hereunder we mostly focus on the reconstruction error case but provide full details on the KL divergence optimization in Appendix A2.

A popular iterative method to approximate the reconstruction error is Lee–Seung's MU scheme (Lee and Seung, 2000):

$$H_{l+1}\leftarrow H_l\odot\frac{W_l^TV}{W_l^TW_lH_l} \tag{1}$$

$$W_{l+1}\leftarrow W_l\odot\frac{VH_l^T}{W_lH_lH_l^T}, \tag{2}$$

where $\odot$, $\frac{[\cdot]}{[\cdot]}$ denote entry-wise multiplication and division, superscript T denotes matrix transpose, and the subscript index denotes the iteration number. Usually, W_0, H_0 are initialized by random or fixed non-negative values; more complicated initialization strategies have also been introduced (Albright et al., 2006; Boutsidis and Gallopoulos, 2008).

2.1.1. Regularized variants. Hoyer (2002) extended the classical MU scheme for the case of an L_1 penalty imposed on the coefficients of H. Other works have also developed formulations for L_2 regularization (Wang et al., 2016). For completeness, we redevelop a regularized variant with both penalties in Appendix A1. Fixing W and looking at one sample v and one column h of H at a time, we consider the problem:

$$\min_{h\geq 0}\left\{\frac{1}{2}\|v-Wh\|_2+\lambda_1\|h\|_1+\frac{1}{2}\lambda_2\|h\|_2^2\right\}. \tag{3}$$

This leads to the following MU equation (see Appendix A1):

$$h_{l+1}\leftarrow h_l\odot\frac{W^Tv}{W^TWh_l+\lambda_1+\lambda_2h_l}. \tag{4}$$

Note that if h_0, W, v and the regularization parameters λ_1, λ_2 are non-negative, then h_l will be non-negative as well.

2.2. Unrolling the iterative algorithm

To obtain our suggested unrolled network, it will be convenient to consider one input sample $v\in R^f$ at a time. Following Hershey et al. (2014), we develop the network architecture by optimizing the corresponding column h while allowing W to be part of the network's parameters that are being learned and,

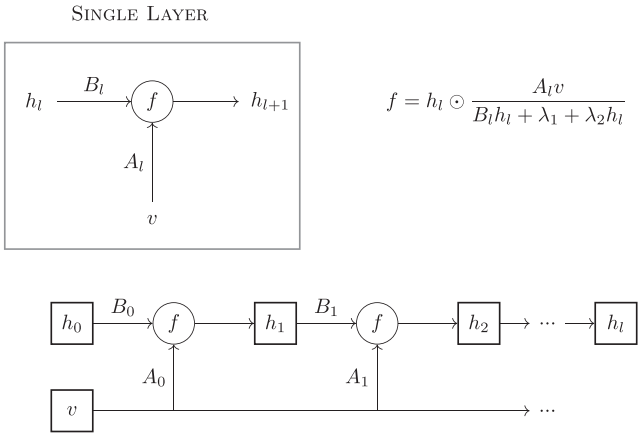

FIG. 1. A sketch of the proposed supervised unrolled network for non-negative matrix factorization.

moreover, vary between layers. In the unrolled network, each layer represents a possible solution to h that is formed by a nonlinear transformation of the values at the previous layer. The transformation imitates the MU Eq. (4), with W varying between the layers (rather than being fixed) and λ_1, λ_2 fixed across layers. Moreover, the network ignores the dependency between the W^T term and the $W^T W$ terms in the updated formula and treats them as independent matrices, A and B, respectively. These matrices are later learned from data. Overall, in the supervised setting, the network relies on training data $v_1, v_2, \ldots v_N \in R^f$ and their corresponding coefficient vectors $h'_1, h'_2, \ldots h'_N \in R^k$ to optimize the parameters $A_l, B_l, \lambda_1, \lambda_2$. The resulting network model is depicted in Figure 1. We note that the L_2 regularization term $\lambda_2 h_l$ could be combined into $B_l h_l$, thus simplifying the update function f in each layer, but in practice separating the two terms leads to better results.

Notably, a similar unrolling architecture applies to the KL divergence case with the main differences being the loss function used and the MU function, which involves two learned matrices that represent W and W^T in the update formula (see Appendix A2).

To test the resulting network, we used 10 layers (see Section 3 for performance across varying depth values) and implemented backpropagation using Pytorch. Training was performed through minimizing the MSE loss function $\|h_{10} - h'\|_2$ using the ADAM optimizer with learning rate 0.001. The parameters were updated using constrained gradient descent to guarantee that network weights are non-negative.

The model parameters including the A, B matrices across layers and the two regularization parameters λ_1, λ_2 were initialized to a fixed positive value (value of 1). We also initialized the entries of h_0 to the same

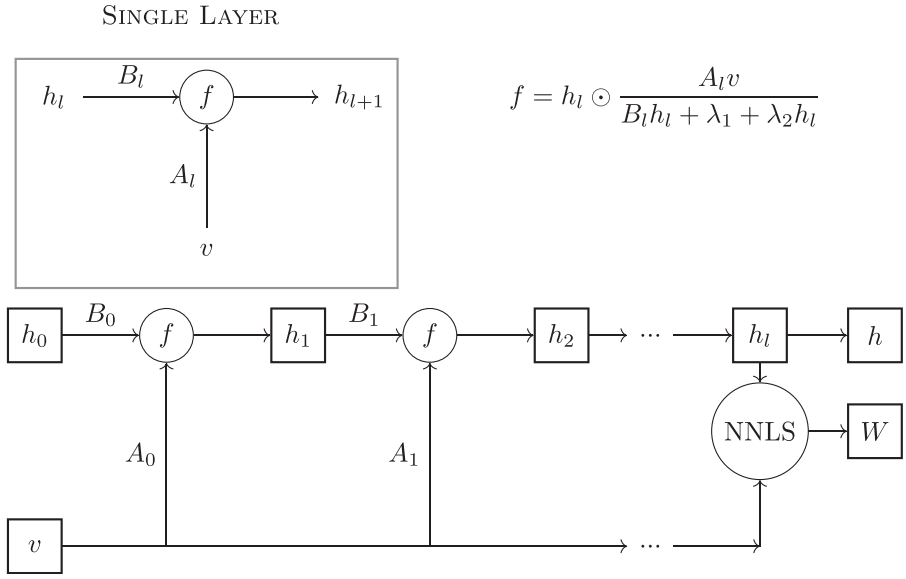

FIG. 2. A sketch of the unrolled deep network for the unsupervised variant. NNLS, non-negative least squares.

value. For each of the data sets we trained a model based on 80% of the data, and measured the MSE with respect to the remaining 20% using the true matrix H.

2.3. An unsupervised variant

Typically, we do not know the decomposition matrices H and/or W in advance, in which case supervised training is not feasible. Instead, we propose to evaluate a solution by its ability to reconstruct the original matrix V. To this end, after obtaining the network output h for each of the data columns, we use non-negative least squares (NNLS) to reconstruct W (Lawson and Hanson, 1995) and adjust the cost function accordingly. In detail, we start by initializing h_0 to fixed values for every column of V, the two columns are forward propagated in the network, and the resulting h_ℓ-s for all samples are gathered to form the estimated H matrix. Next, we apply NNLS to estimate W from V and H. Last, we calculate the cost function given in Eq. (3) and backpropagate to update the network weights A, B. The model is depicted in Figure 2.

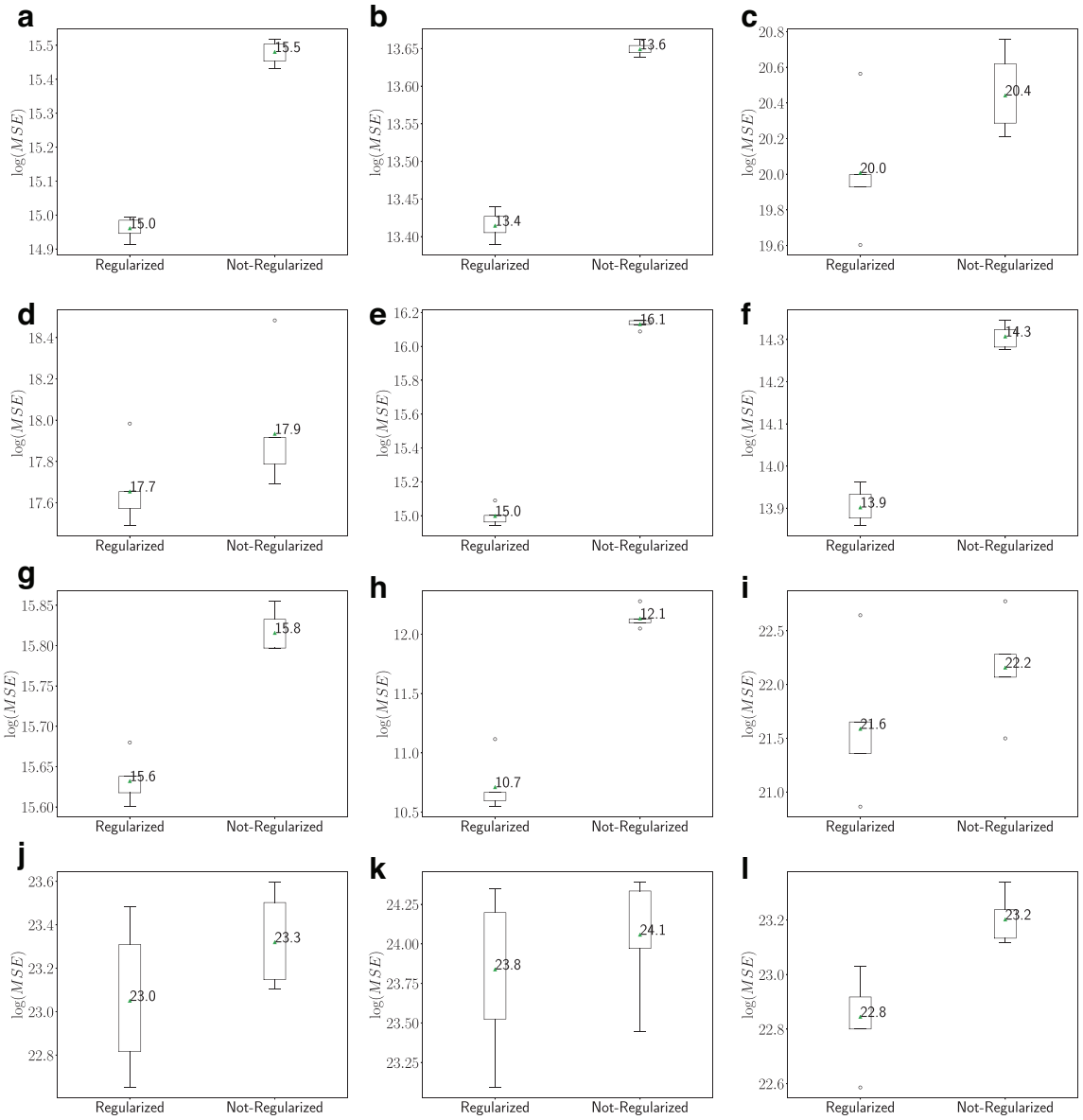

FIG. 3. The effect of regularization on DNMF performance in the supervised case. **(a–l)** Represent simulated data sets (1–12), respectively. DNMF, deep non-negative matrix factorization.

In the unsupervised case we cannot learn the regularization parameters as they affect the cost function and if we would omit them from the cost function, their optimal value will be zero (corresponding to no regularization). Hence, in this variant we use $\lambda_1 = \lambda_2 = \lambda$ and present results for $\lambda = 0, 1, 2$.

2.4. Data description and performance evaluation

We used two types of mutation data sets: simulated and real ones. In all cases the number of rows in the observed (count) matrix V was 96, representing the 96 standard mutation categories (Alexandrov et al., 2019). For such data, V is assumed to be the result of the activity of certain mutational processes whose signatures are given by the dictionary W and whose exposures are given by the coefficient matrix H. We describe these data sets as follows.

2.4.1. Simulated data. The simulated data were taken from Alexandrov et al. (2019) and includes multiple mutation data sets with varying numbers of underlying signatures and degrees of noise. For each data set we are given an observed matrix of mutation counts (denoted V earlier) and its decomposition into signature (W) and exposure (H) matrices. In total, we used 12 different simulated data sets with at least 1000 samples each as detailed in Appendix A3.

2.4.2. Real data. We analyzed a breast cancer (BRCA) mutation data set of whole-genome sequences from the International Cancer Genome Consortium. The data set has 560 samples and believed to be the result of the activity of 12 mutational processes as cataloged in the Catalogue of Somatic Mutations in Cancer (COSMIC) database.

2.4.3. Performance evaluation. We evaluated our method on each data set using fivefold cross-validation and compared with the standard MU method under various regularization schemes. All model parameters in both methods were initialized to one, unless specified otherwise. In the supervised case, we report the MSE between the true H and the estimated matrix H_l over a test set (20% of the samples), where the MSE is averaged over the columns of H. In the unsupervised case, we report the MSE between V and its reconstruction WH over a test set (20% of the samples), where the MSE is averaged over the columns of H. For both DNMF and MU, W in inferred using the training samples and H is estimated on the test samples. For DNMF, the estimation of H is done by propagating the columns of V in the learned network. For MU, it is done by fixing W and using the iterative update rule to compute H.

2.5. Implementation and runtime details

All reported runs were done in Google Colab using a 2-core CPU (x86_64). Code is available at https://github.com/raminass/deep-NMF. The inference time of supervised/unsupervised DNMF with 10 layers was 0.0019 seconds, similar to a 10-iteration MU inference in the supervised case (0.0021 seconds), and an order of magnitude faster than a 100-iteration MU inference in the unsupervised case (as used in this study, 0.016 seconds).

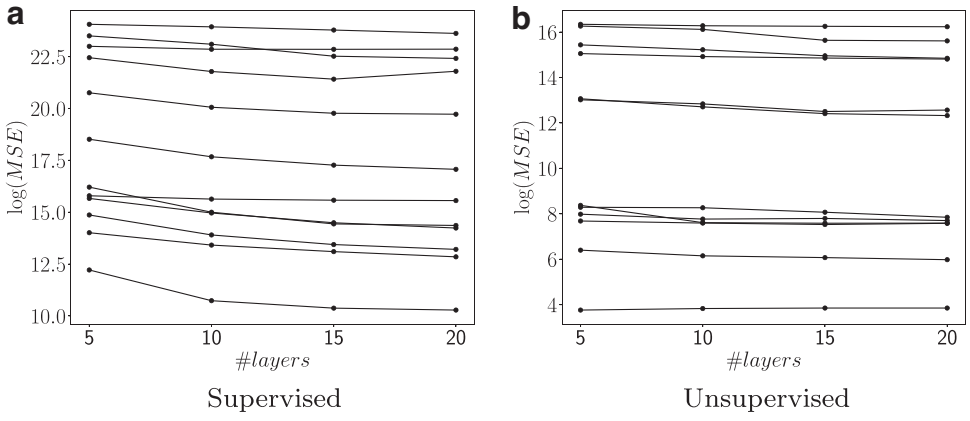

FIG. 4. The effect of number of layers on algorithm's performance. Each line corresponds to one of the simulated data sets. **(a)** Supervised; **(b)** unsupervised.

3. RESULTS

We developed a deep learning based framework for NMF, which we call DNMF. The DNMF framework imitates the classical MU scheme for the problem by unrolling its iterations as layers in a deep network model. We further developed regularized variants for MU and DNMF. We apply our framework in both a supervised setting, where training data regarding the true factorization is available, and in an unsupervised setting. Full details on the different models appear in Section 2.

We start with testing the different model formulations using simulated data. First, we compare the regularized with the nonregularized variant in the supervised case. As expected, the results, summarized in Figure 3, show that the regularized variant performs best in terms of MSE, hence we focus on it in the sequel.

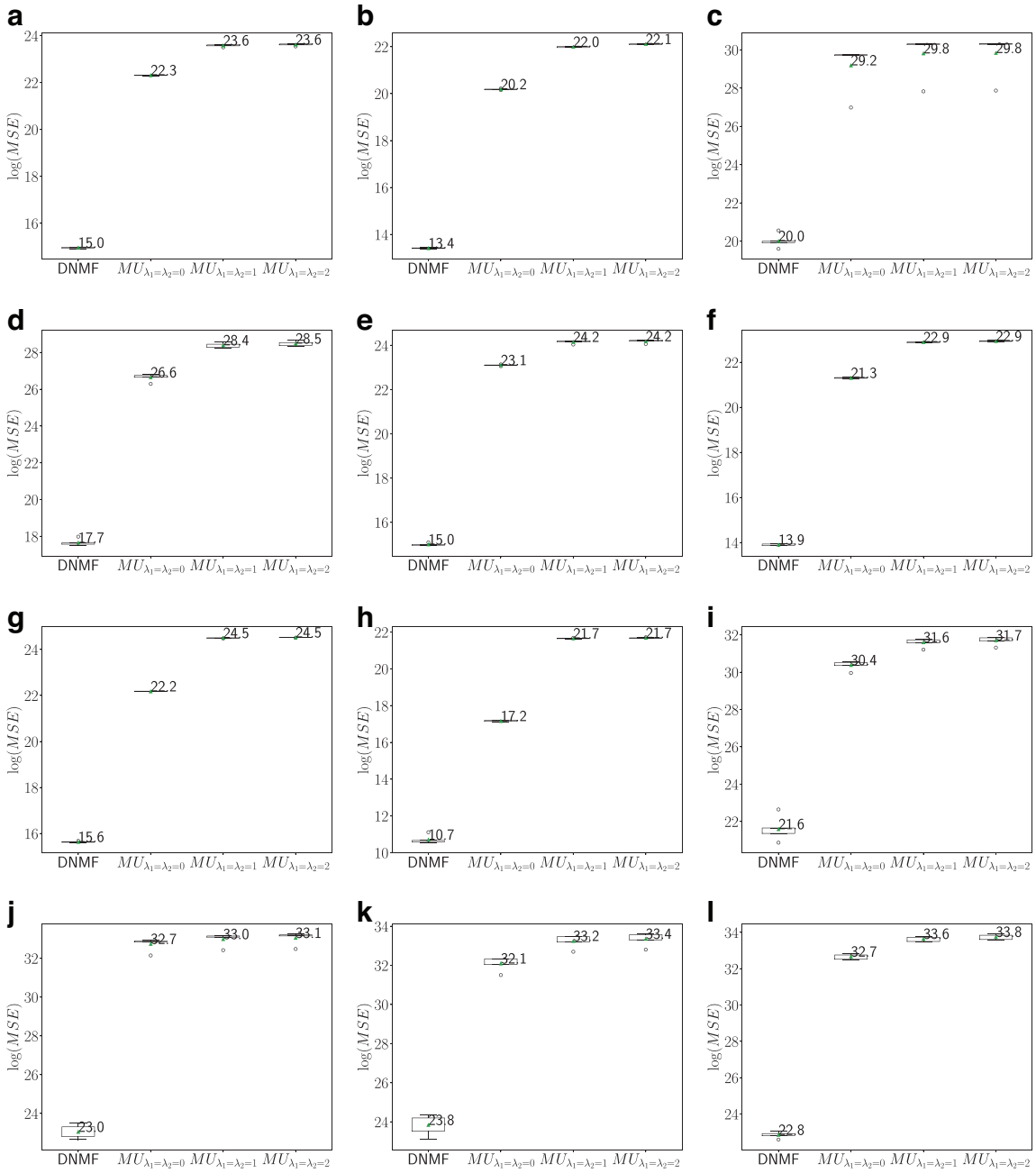

FIG. 5. Comparative performance on simulated data in the supervised setting. **(a–l)** Represent simulated data sets (1–12), respectively.

Next, we tested the effect of the depth of the unrolled network on the algorithm's performance. The results, depicted in Figure 4, show that after 10–15 layers the performance reaches a plateau, hence we focus in the following on depth-10 networks. Notably, as our network borrows from the MU update scheme and does not rely on activation functions, it is less affected by the problem of gradient decay for deep architectures.

After determining the architecture of the developed framework, we turn to examine it in the supervised case and compare with the MU approach on the simulated data. To this end, we apply MU to the training data to estimate W, and then use MU with the learned W to estimate H on the test data. The results, summarized in Figure 5, show that DNMF outperforms MU across a wide range of regularization values for the latter (note that DNMF learns the regularization parameters automatically from data in this case).

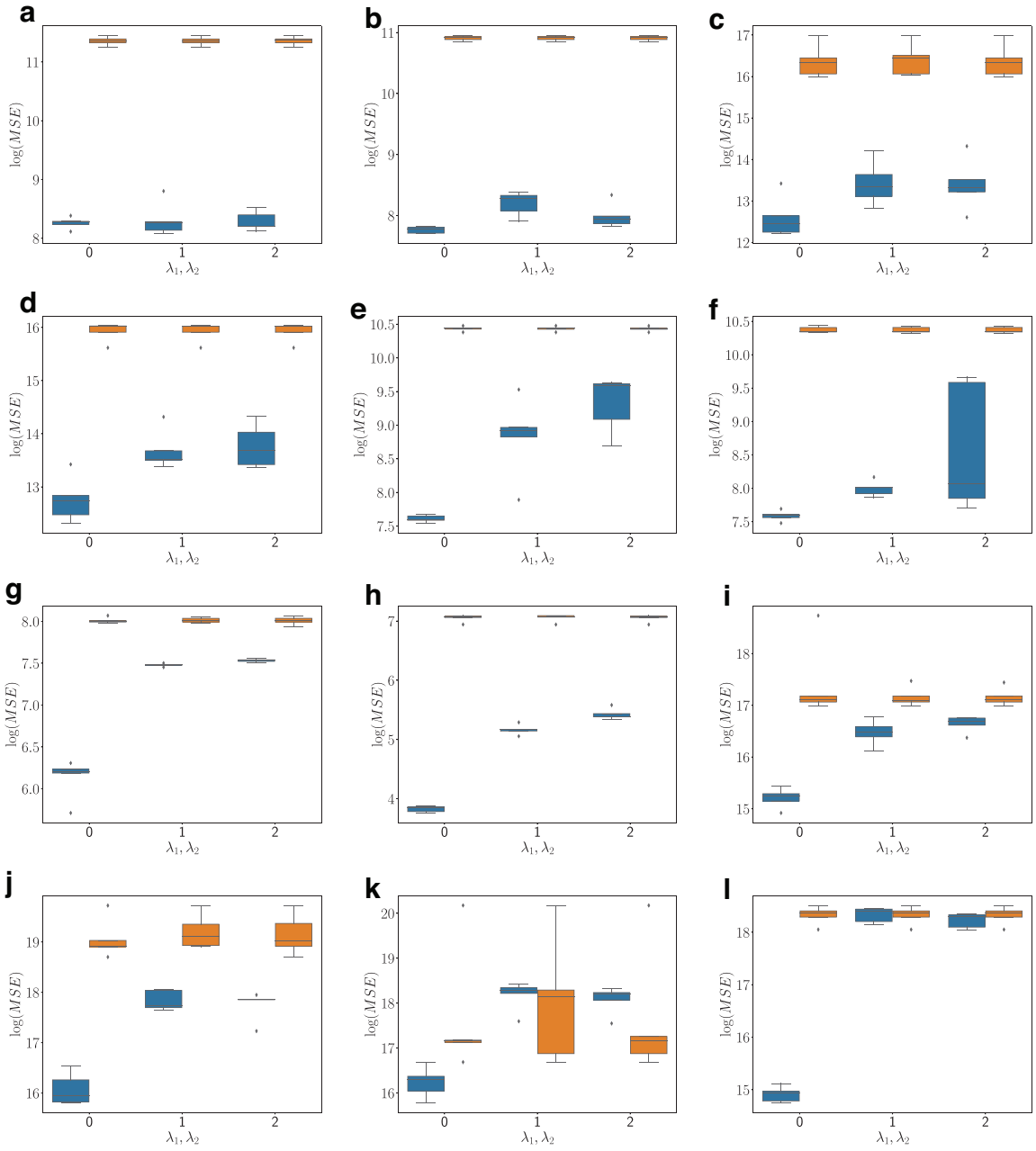

FIG. 6. Comparative performance in the unsupervised setting on simulated data. Blue: DNMF; orange: MU. **(a–l)** Represent simulated data sets (1–12), respectively. MU, multiplicative update.

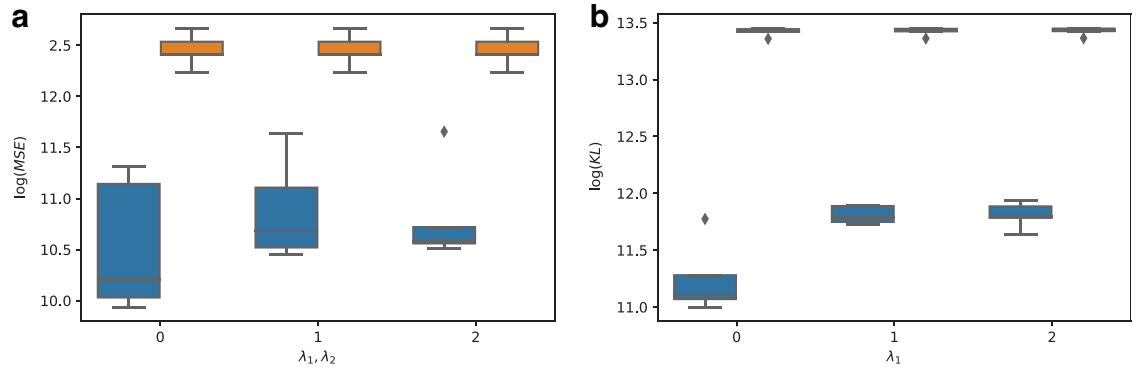

FIG. 7. Comparative performance in the unsupervised setting on real data for both Euclidean **(a)** and Kullback–Leibler divergence **(b)** objectives. Blue: DNMF; orange: MU.

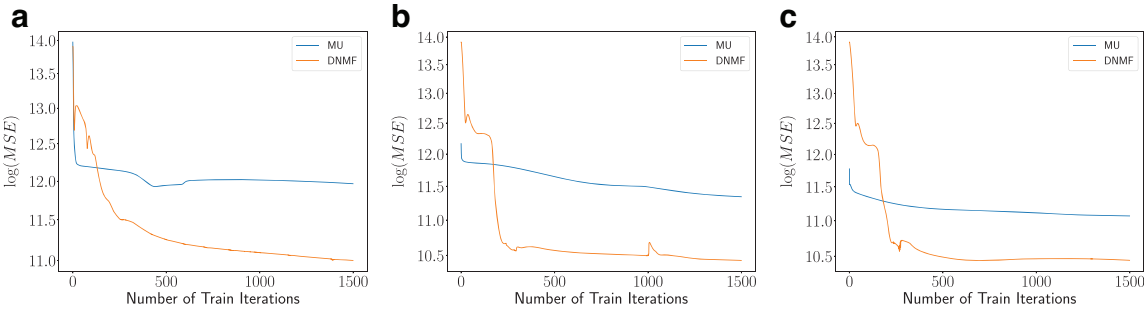

FIG. 8. Unsupervised reconstruction error during training on real data for MU and DNMF. Values of λ_1 and λ_2 from left to right: **(a)** $\lambda_1 = \lambda_2 = 0$; **(b)** $\lambda_1 = \lambda_2 = 1$; **(c)** $\lambda_1 = \lambda_2 = 2$.

Next, we evaluate DNMF in the unsupervised case. In this case, the regularization parameters are part of the objective function and cannot be learned by the model, hence we compare DNMF with MU under different regularization settings. As evident from the results in Figures 6 and 7a, DNMF outperforms MU across a wide range of data sets and regularization values on both simulated and real data.

For the real mutation data, we also evaluate the KL divergence variant of DNMF as this type of optimization is commonly applied to such data (Kim et al., 2016). The results, depicted in Figure 7b, again show the superiority of DNMF over MU.

To get an intuition for the improved performance of DNMF compared with MU, we looked at the cost function being optimized across algorithmic iterations when considering the real data set and multiple regularization parameters. We observed that MU converges to a local minimum after a few iterations only, hence we attempted different random initializations for it and report the best one. Nevertheless, DNMF remains the best performer under all settings (Fig. 8).

4. CONCLUSIONS

We provided a detailed deep learning framework for NMF that is applicable in both supervised and unsupervised settings. The framework outperforms classical approaches to this problem and greatly improves the reconstruction error of the factorization across a wide range of data sets and regularization schemes. We demonstrated the utility of our framework in analyzing mutation data from simulated and real data sets and expect it to greatly improve our ability to reconstruct mutational signatures and their exposures.

For future study, we intend to explore different strategies for initializing the DNMF model and to select its regularization parameters in the unsupervised case.

ACKNOWLEDGMENTS

We thank Itay Sason for his helpful feedback on the article.

AUTHOR DISCLOSURE STATEMENT

The authors declare they have no conflicting financial interests.

FUNDING INFORMATION

R.S. was supported by a grant from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel.

REFERENCES

Albright, R., Cox, J., Duling, D., et al. 2006. Algorithms, initializations, and convergence for the nonnegative matrix factorization. *Tech. Rep. Citeseer*.

Alexandrov, L.B., Kim, J., Haradhvala, N.J., et al. 2019. The repertoire of mutational signatures in human cancer. *Nature* 578, 94–101.

Boutsidis, C., and Gallopoulos, E. 2008. SVD based initialization: A head start for nonnegative matrix factorization. *Pattern Recognit*. 41, 1350–1362.

Gillis, N. 2014. The why and how of nonnegative matrix factorization, 257–291. *In* Suykens, J.A.K., Signoretto, M., and Argyriou, A., eds. *Regularization, Optimization, Kernels, and Support Vector Machines*.

Hershey, J.R., Roux, J.L., and Weninger, F. 2014. Deep unfolding: Model-based inspiration of novel deep architectures. *Mach. Learn.* 4, 1–27.

Hoyer, P.O. 2002. Non-negative sparse coding, 557–565. *In Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*. IEEE, Martigny, Switzerland.

Kim, J., Mouw, K.W., Polak, P., et al. 2016. Somatic ERCC2 mutations are associated with a distinct genomic signature in urothelial tumors. *Nat. Genet*. 48, 600–606.

Kim, J., and Park, H. 2011. Fast nonnegative matrix factorization: An active-set-like method and comparisons. *SIAM J. Sci. Comput*. 33, 3261–3281.

Lawson, C.L., and Hanson, R.J. 1995. *Solving Least Squares Problems*. SIAM.

Lee, D.D., and Seung, H.S. 2000. Algorithms for non-negative matrix factorization, 535–541. *In Proceedings of the 13th International Conference on Neural Information Processing Systems* (NIPS'00). Denver, CO: MIT Press.

Lin, C.-J. 2007. Projected gradient methods for nonnegative matrix factorization. *Neural Comput*. 19, 2756–2779.

Monga, V., Li, Y., and Eldar, Y.C. 2021. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Sign. Process. Magaz.* 38, 18–44.

Vavasis, S.A. 2010. On the complexity of nonnegative matrix factorization. *SIAM J. Optim.* 20, 1364–1377.

Wang, D., Liu, J.-X., Gao, Y.-L., et al. 2016. An NMF-L2,1-norm constraint method for characteristic gene selection. *PLoS One* 11, e0158494.

Wisdom, S., Powers, T., Pitton, J., et al. 2017. Building recurrent networks by unfolding iterative thresholding for sequential sparse recovery. *Proc. IEEE Int. Conf. Acoust. Speech Signal Process*. 4346–4350.

Address correspondence to:
Prof. Roded Sharan
Blavatnik School of Computer Science
Tel Aviv University
Tel Aviv 69978
Israel

E-mail: roded@tauex.tau.ac.il

(*Appendix follows* →)

Appendix

APPENDIX A1. REGULARIZED NON-NEGATIVE MATRIX FACTORIZATION

Consider the problem of finding an approximate non-negative factorization that is close to the original matrix V and satisfies the sparseness constrains. We use the Frobenius norm as a measure of the distance between V and WH, adding L_1, L_2 regularizations, arriving at the following cost function:

$$C(W,H)=\frac{1}{2}\,\|V-WH\|_F^2\;+\lambda_1\;\|H\|_1+\frac{1}{2}\lambda_2\;\|H\|_2^2\;. \tag{A1}$$

Theorem A1. *The cost function is nonincreasing under the update rules:*

$$H\leftarrow H\odot\frac{W^TV}{W^TWH+\lambda_1+\lambda_2H}\quad;\quad W\leftarrow W\odot\frac{VH^T}{WHH^T}.$$

Proof. We follow the proofs of Lee and Seung (2000; Hoyer, 2002) and focus on the update formula for H, considering one column h at a time corresponding to a column v of V. Our goal is to minimize

$$C(h)=\frac{1}{2}\,\|v-Wh\|_F^2\;+\lambda_1\|h\|_1+\frac{\lambda_2}{2}\|h\|_2^2\;.$$

As in these references, we define $G(h,h_l)$ to be an auxiliary function for $C(h)$ that satisfies $G(h,h_l)\geq C(h)$, $G(h,h)=C(h)$. At each iteration we update as follows:

$$h_{l+1}=\underset{\mathrm{h}}{\operatorname{argmin}}\;G(h,h_l).$$

We keep the original definition of the auxiliary function:

$$G(h,h_l)=C(h_l)+(h-h_l)^T\nabla C(h_l)+\frac{1}{2}(h-h_l)^TK(h_l)(h-h_l),$$

where $K(h_l)$ is a diagonal matrix. However, we slightly change $K(h_l)$ to reflect the regularization: $K_{ab}(h_l):=K'_{ab}(h_l)+\lambda_2=\delta_{ab}\frac{(W^TWh_l)_a+\lambda_1}{h_{la}}+\lambda_2$. To show that $G(h,h_l)\geq C(h)$ we take a Taylor expansion of C:

$$C(h)=C(h_l)+(h-h_l)^T\nabla C(h_l)+\frac{1}{2}(h-h_l)^T(W^TW+\lambda_2)(h-h_l).$$

Thus, we need to show that $0\leq(h-h_l)^T(K'(h_l)-W^TW)(h-h_l)$, which was shown in Hoyer (2002).

It remains to compute the gradient of G and equate it to zero:

$$\nabla_hG(h,h_l)=\nabla C(h_l)+(h-h_l)K(h_l)=0.$$

This gives the update rule $h_{l+1}=h_l-K(h_l)^{-1}\nabla C(h_l)$ where $\nabla C(h_l)=-W^Tv+W^TWh_l+\lambda_2h_l+\lambda_1$. Overall, we get

$$h_{l+1}=h_l-\frac{h_l}{W^TWh_l+\lambda_1+\lambda_2h_l}(-W^Tv+W^TWh_l+\lambda_2h_l+\lambda_1)=h_l\odot\frac{W^Tv}{W^TWh_l+\lambda_1+\lambda_2h_l},$$

completing the proof. □

APPENDIX A2. KULLBACK–LEIBLER DIVERGENCE OPTIMIZATION

We develop update rules to optimize the regularized variant also for the Kullback–Leibler (KL) divergence criterion. The cost function to be optimized is

$$C(W,H)=KL(V\|WH)+\lambda_1\|H\|_1. \tag{A2}$$

(*Appendix continuous*)

Note that in this setting we only use L_1 regularization as the KL divergence does not have a quadratic term.

Theorem A2. *The cost function is nonincreasing under the update rules:*

$$H \leftarrow H \odot \frac{W^T(V \odot (WH)^{-1})}{W^T \mathbf{1}_{f\times n} + \lambda_1} \quad ; \quad W \leftarrow W \odot \frac{(V \odot (WH)^{-1})H^T}{\mathbf{1}_{f\times n} H^T}.$$

Proof. We follow the proof of Lee and Seung (2000) and focus on the update formula for H, considering one column h at a time corresponding to a column v of V. Our goal is to minimize

$$C(h) = KL(v\|Wh) + \lambda_1 \|h\|_1.$$

As before, we keep the original definition of the auxiliary function with the addition of the sparsity factor:

$$G(h, h_l) = G'(h, h_l) + \lambda_1 \|h\|_1.$$

Lee and Seung (2000) proved that $G'(h, h_l) \geq KL(v\|Wh)$, hence $G(h, h_l) \geq C(h)$. Computing the gradient of G and equating it to zero:

$$\nabla_h G(h, h_l) = \nabla_h G'(h, h_l) + \lambda_1 = 0.$$

$$\nabla_h G'(h, h_l) = -\frac{W^T(v \odot (Wh_l)^{-1})}{h} \odot h_l + W^T \mathbf{1}_f.$$

The resulting update to h is

$$h_{l+1} \leftarrow h_l \odot \frac{W^T(v \odot (Wh_l)^{-1})}{W^T \mathbf{1}_f + \lambda_1}.$$

completing the proof. □

APPENDIX A3. SIMULATED DATA

Appendix Table A1. List of Simulated Data Sets Used in This Study

	Data set	*No. samples*	*No. components*
1	pancreas.sp	1000	11
2	pancreas.sa	1000	20
3	many.types.sp	2700	21
4	many.types.sa	2700	39
5	3.5.40.rcc.and.ovary.sp	1000	11
6	3.5.40.rcc.and.ovary.sa	1000	19
7	3.5.40.abst.sp	1000	3
8	3.5.40.abst.sa	1000	3
9	2.7a.7b.bladder.and.melanoma.sp	1000	11
10	2.7a.7b.bladder.and.melanoma.sa	1000	26
11	2.7a.7b.abst.sp	1000	3
12	2.7a.7b.abst.sa	1000	3