Supplementary Materials for GDNMF: Generalized Discriminative Deep Non-Negative Matrix Factorization Based on Latent Feature and Basis Learning

1 Optimization

1.1 Detailed Fine-Tune Process

In the fine-tune stage, we solve the optimization problem of GD^{2}NMF in (1) based on alternating optimization.

$$\begin{aligned}\min_{\substack{\mathbf{F},\mathbf{S},\mathbf{G}\geq 0\\ \mathbf{W},\mathbf{U},\mathbf{V}\geq 0}} &= \frac{1}{2}\|\mathbf{X}-\hat{\mathbf{F}}_1\mathbf{S}_1\hat{\mathbf{G}}_1-\mathbf{W}\mathbf{G}_{m_2}\|_F^2 \\ &+\frac{\alpha}{2}\|(\mathbf{Y}-\mathbf{U}\hat{\mathbf{G}}_{\mathbf{1}}-\mathbf{V}\mathbf{G}_{m_2})\mathbf{Q}\|_F^2, \\ s.t.\quad \hat{\mathbf{F}}_1 &= g(g(g(\mathbf{F}_{m_1}\mathbf{S}_{m_1}^L)\mathbf{S}_{m_1-1}^L)\cdots\mathbf{S}_2^L), \\ \hat{\mathbf{G}}_1 &= g(\mathbf{S}_2^R g(\cdots\mathbf{S}_{m_2-1}^R g(\mathbf{S}_{m_2}^R\mathbf{G}_{m_2}))),\end{aligned} \tag{1}$$

Let $\mathcal{O}$ denote the objective value in (1). The algorithm repeats the following steps until convergence.

1.1.1 Update F

To optimize (1) over $\mathbf{F}$, we solve the following sub-problem:

$$\min_{\mathbf{F}\geq 0}\frac{1}{2}\|\mathbf{X}-\hat{\mathbf{F}}_1\mathbf{S}_1\hat{\mathbf{G}}_1-\mathbf{W}\mathbf{G}_{m_2}\|_F^2 \tag{2}$$

Based on the chain rule of derivatives, we get the gradient w.r.t. $\mathbf{F}$ for the i-th layer,

$$\begin{aligned}\nabla_{\mathbf{F}_i}\mathcal{O} &= \frac{\partial\mathcal{O}}{\partial\mathbf{F}_i} = \frac{\partial\mathcal{O}}{\partial\mathbf{F}_i\mathbf{S}_i^L}(\mathbf{S}_i^L)^T \\ &= \left[\frac{\partial\mathcal{O}}{\partial g(\mathbf{F}_i\mathbf{S}_i^L)}\odot g^{'}(\mathbf{F}_i\mathbf{S}_i^L)\right](\mathbf{S}_i^L)^T \\ &= \left[\nabla_{\mathbf{F}_{i-1}}\mathcal{O}\odot g^{'}(\mathbf{F}_i\mathbf{S}_i^L)\right](\mathbf{S}_i^L)^T,\end{aligned} \tag{3}$$

and the derivative for $\mathbf{F}_1$ is :

$$\nabla_{\mathbf{F}_1}\mathcal{O} = -(\mathbf{X} - \mathbf{F}_1\mathbf{S}_1\mathbf{G}_1 - \mathbf{W}\mathbf{G}_{m_2})\mathbf{G}_1^T\mathbf{S}_1^T. \tag{4}$$

1.1.2 Update S

Similarly, we can get the gradient w.r.t. $\mathbf{S}_i^L$ and $\mathbf{S}_i^R$ based on the chain rule,

$$\begin{aligned}\nabla_{\mathbf{S}_i^L}\mathcal{O} &= \frac{\partial\mathcal{O}}{\partial\mathbf{S}_i^L} = \mathbf{F}_i^T\frac{\partial\mathcal{O}}{\partial\mathbf{F}_i\mathbf{S}_i^L} \\ &= \mathbf{F}_i^T\left[\frac{\partial\mathcal{O}}{\partial g(\mathbf{F}_i\mathbf{S}_i^L)} \odot g^{'}(\mathbf{F}_i\mathbf{S}_i^L)\right] \\ &= \mathbf{F}_i^T\left[\nabla_{\mathbf{F}_{i-1}}\mathcal{O} \odot g^{'}(\mathbf{F}_i\mathbf{S}_i^L)\right],\end{aligned} \tag{5}$$

$$\begin{aligned}\nabla_{\mathbf{S}_i^R}\mathcal{O} &= \frac{\partial\mathcal{O}}{\partial\mathbf{S}_i^R} = \frac{\partial\mathcal{O}}{\partial\mathbf{S}_i^R\mathbf{G}_i}\mathbf{G}_i^T \\ &= \left[\frac{\partial\mathcal{O}}{\partial g(\mathbf{S}_i^R\mathbf{G}_i)} \odot g^{'}(\mathbf{S}_i^R\mathbf{G}_i)\right]\mathbf{G}_i^T \\ &= \left[\nabla_{\mathbf{G}_{i-1}}\mathcal{O} \odot g^{'}(\mathbf{S}_i^R\mathbf{G}_i)\right]\mathbf{G}_i^T.\end{aligned} \tag{6}$$

The derivative for $\mathbf{S}_1$ is:

$$\nabla_{\mathbf{S}_1}\mathcal{O} = -\mathbf{F}_1^T(\mathbf{X} - \mathbf{F}_1\mathbf{S}_1\mathbf{G}_1 - \mathbf{W}\mathbf{G}_{m_2})\mathbf{G}_1^T. \tag{7}$$

1.1.3 Update G

The gradient w.r.t. $\mathbf{G}_i$ is obtained as,
if $i = 1$:

$$\begin{aligned}\nabla_{\mathbf{G}_1}\mathcal{O} = &-(\mathbf{F}_1\mathbf{S}_1)^T(\mathbf{X} - \mathbf{F}_1\mathbf{S}_1\mathbf{G}_1 - \mathbf{W}\mathbf{G}_{m_2}) \\ &- \alpha\mathbf{P}_1^T(\mathbf{Y} - \mathbf{U}\mathbf{G}_1 - \mathbf{V}\mathbf{G}_{m_2})\mathbf{Q}\mathbf{Q}^T,\end{aligned} \tag{8}$$

if $1 < i < m_2$:

$$\begin{aligned}\nabla_{\mathbf{G}_i}\mathcal{O} &= \frac{\partial\mathcal{O}}{\partial\mathbf{G}_i} = (\mathbf{S}_i^R)^T\frac{\partial\mathcal{O}}{\partial\mathbf{S}_i^R\mathbf{G}_i} \\ &= (\mathbf{S}_i^R)^T\left[\frac{\partial\mathcal{O}}{\partial g(\mathbf{S}_i^R\mathbf{G}_i)} \odot g^{'}(\mathbf{S}_i^R\mathbf{G}_i)\right] \\ &= (\mathbf{S}_i^R)^T\left[\nabla_{\mathbf{G}_{i-1}}\mathcal{O} \odot g^{'}(\mathbf{S}_i^R\mathbf{G}_i)\right],\end{aligned} \tag{9}$$

if $i = m_2$:

$$\begin{aligned}\nabla_{\mathbf{G}_{m_2}}\mathcal{O} = &(\mathbf{S}_{m_2}^R)^T\left[\nabla_{\mathbf{G}_{m_2-1}}\mathcal{O} \odot g^{'}(\mathbf{S}_{m_2}^R\mathbf{G}_{m_2})\right] \\ &- \mathbf{W}^T(\mathbf{X} - \mathbf{F}_1\mathbf{S}_1\mathbf{G}_1 - \mathbf{W}\mathbf{G}_{m_2}) \\ &- \alpha\mathbf{P}^T(\mathbf{Y} - \mathbf{U}\mathbf{G}_1 - \mathbf{V}\mathbf{G}_{m_2})\mathbf{Q}\mathbf{Q}^T.\end{aligned} \tag{10}$$

Algorithm 1 Pseudocode of GD^{2}NMF

Input: data matrix $\mathbf{X} \in \mathcal{R}^{p \times n}$, label information $\mathbf{y} \in \mathcal{R}^{n \times 1}$, hyperparameters α and r
Output: feature matrices $\{\mathbf{G}_i\}$

Pre-Training:
for all layers **do**
  Initialize $\mathbf{F}_i$ based on (12).
  Initialize $\mathbf{G}_i$ based on (16).
  Initialize $\mathbf{S}_i$ based on (13) and (15).
end for
Fine-Tune:
repeat
  for all layers **do**
    Update $\mathbf{F}_i$ based on (18) and (19).
    Update $\mathbf{G}_i$ based on (23).
    Update $\mathbf{S}_i$ based on (20), (21) and (22).
    Update $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{W}$ based on (24), (25) and (26), respectively.
  end for
until Convergence

These gradients are propagated backwards in the multi-layer model, and in a specific layer, gradient descent is used to update the layer weights once gradients from the previous layer are received.

1.1.4 Update W, V and U

$$\nabla_{\mathbf{W}}\mathcal{O} = -(\mathbf{X} - \mathbf{F}_1\mathbf{S}_1\mathbf{G}_1 - \mathbf{W}\mathbf{G}_{m_2})\mathbf{G}_{m_2}^T, \tag{11}$$

$$\nabla_{\mathbf{U}}\mathcal{O} = -\alpha(\mathbf{Y} - \mathbf{U}\mathbf{G}_1 - \mathbf{V}\mathbf{G}_{m_2})\mathbf{Q}\mathbf{Q}^T\mathbf{G}_1^T, \tag{12}$$

$$\nabla_{\mathbf{V}}\mathcal{O} = -\alpha(\mathbf{Y} - \mathbf{U}\mathbf{G}_1 - \mathbf{V}\mathbf{G}_{m_2})\mathbf{Q}\mathbf{Q}^T\mathbf{G}_{m_2}^T. \tag{13}$$

Using the gradient descent method, we derive the multiplicative update rules as follows:

$$\mathbf{W} = \mathbf{W} \odot \frac{\mathbf{X}\mathbf{G}_{m_2}^T}{(\mathbf{F}_1\mathbf{S}_1\mathbf{G}_1 + \mathbf{W}\mathbf{G}_{m_2})\mathbf{G}_{m_2}^T}, \tag{14}$$

$$\mathbf{U} = \mathbf{U} \odot \frac{\alpha\mathbf{Y}\mathbf{Q}\mathbf{Q}^T\mathbf{G}_1^T}{\alpha(\mathbf{U}\mathbf{G}_1 + \mathbf{V}\mathbf{G}_{m_2})\mathbf{Q}\mathbf{Q}^T\mathbf{G}_1^T}, \tag{15}$$

$$\mathbf{V} = \mathbf{V} \odot \frac{\alpha\mathbf{Y}\mathbf{Q}\mathbf{Q}^T\mathbf{G}_{m_2}^T}{\alpha(\mathbf{U}\mathbf{G}_1 + \mathbf{V}\mathbf{G}_{m_2})\mathbf{Q}\mathbf{Q}^T\mathbf{G}_{m_2}^T}. \tag{16}$$

1.2 Pseudocode

2 Details of Datasets

Five real-world datasets are used in the experiments.

ARface database[1]. There are 3120 facial images of 120 people. Each person has 26 images and Each image contains 928 pixels.

CMUPIE database[2]. There are 2856 facial images from 68 people. Each person has 42 images and the resolution of each image is 32×32.

Yale Face database[3]. There are 165 facial images from 15 people. Each person has 11 images and the resolution of each image is 32×32.

Caltech101-20 database[4]. There are 2386 images from 20 classes. Each image contains 928 pixels.

COIL20 database[5]. There are 1440 digital images from 20 subjects. Each subject has 72 images and the resolution of each image is 32×32.

3 More Experiment Results

3.1 Ablation Study

To evaluate the effectiveness of deep feature and basis decomposition in GD^{2}NMF, we conduct an experiment to compare the clustering results of GD^{2}NMF with its two degraded variants: GD^{2}NMF with only basis decomposition (GD^{2}NMF$_b$) and GD^{2}NMF with only feature decomposition (GD^{2}NMF$_f$). The experiment is conducted on the CMUPIE, Yale face and COIL20 datasets, and result in ACC is shown in Fig.1(a). For fair comparison, the hyperparameters are set to the same values. From Fig.1(a), we can see that GD^{2}NMF achieves better performance than both GD^{2}NMF$_b$ and GD^{2}NMF$_f$ on the three datasets, which validates our assumption that simultaneously performing deep factorization on features and bases leads to a improved data representation and clustering performance.

To verify the effectiveness of incorporating deep and shallow models in GD^{2}NMF, we conducted an experiment to compare the performance of GD^{2}NMF, GD^{2}NMF in shallow architecture (GD^{2}NMF$_s$) and GD^{2}NMF in deep architecture (GD^{2}NMF$_d$). The experiments are conducted on the CMUPIE dataset. In the experiment, we extract different number of classes from the original CMUPIE dataset, and generate six datasets by varying number of classes from 20 to 60 by step 10. Results in ACC is shown in Fig.1(b). GD^{2}NMF usually achieves a better performance than its two variants, which verifies our assumption that mutually reinforcing shallow model and deep model helps to boost the generalization performance.

[1] https://www2.ece.ohio-state.edu/ aleix/ARdatabase.html

[2] https://www.cs.cmu.edu/afs/cs/project/PIE/MultiPie/Multi-Pie/Home.html

[3] http://cvc.cs.yale.edu/cvc/projects/yalefaces/yalefaces.html

[4] http://www.vision.caltech.edu/Image Datasets/Caltech101/

[5] https://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php

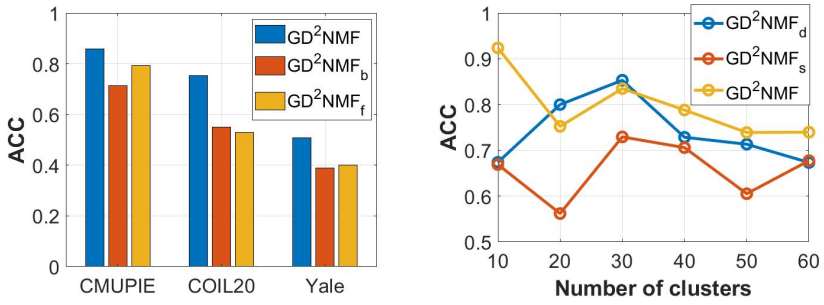


Figure R1: (a): Comparison of GD^{2}NMF, GD^{2}NMF$_b$ and GD^{2}NMF$_f$ on three datasets. GD^{2}NMF$_b$ and GD^{2}NMF$_f$ only consider basis decomposition and feature decomposition, respectively. (b): Comparison of GD^{2}NMF, GD^{2}NMF$_d$ and GD^{2}NMF$_s$ on CMUPIE. GD^{2}NMF$_d$ and GD^{2}NMF$_s$ are variants of GD^{2}NMF by ignoring the shallow model and the deep model, respectively.

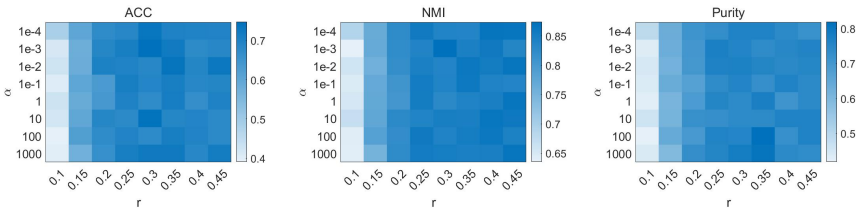


Figure R2: Hyperparameter sensitivity analysis of α and r in terms of ACC, NMI and Purity on the CMUPIE dataset.

3.2 Hyperparameter Sensitivity Analysis

There are two hyperparameters in GD^{2}NMF. The decay rate r is the ratio of the dimensions between successive layers, and α controls the weight of soft label constraint. In experiments, the value of r is selected from $\{0.1, 0.15, ..., 0.45\}$, and the value of α is selected from $\{10^{-4}, 10^{-3}, ..., 10^{3}\}$. The experiment is conducted on the CMUPIE dataset and the clustering results are shown in Fig.R2. It shows that GD^{2}NMF is less sensitive to α on the CMUPIE dataset, and it achieves stable performance once $r > 0.15$. The results indicate that GD^{2}NMF is stable and robust in terms of the two hyperparameters.

3.3 Runtime Analysis

Experiment was conducted on a desktop computer configured with Intel(R) Core(TM) i7-10700 CPU@2.90GHz and matlab R2020b. We conduct the experiment on the CMUPIE dataset for runtime analysis with six comparison methods (CNMF [2], DNBMF

[3], DSNMF [4], Deep-WSF [4], SGDNMF [1]), and results are shown in Table R1.

Table R1: Runtime (in seconds) analysis for six comparison methods.

	CNMF	DNBMF	DSNMF	Deep-WSF	SGDNMF	GD^{2}NMF
CMUPIE	20.95	101.44	199.82	251.39	886.15	236.38
COIL20	5.82	100.04	105.97	126.49	305.85	61.95
ARface	20.70	133.14	124.09	346.12	1332.86	156.88
Caltech20	9.88	79.35	156.95	208.77	719.31	120.64
Yale	1.52	5.46	32	36.29	15.47	3.91

3.4 Soft Constraints vs Hard Constraints

SGDNMF [1] utilizes the dual-hypergraph Laplacian regularization to capture the high-order relations among data points and enforce the partial label information via a label constraint matrix. For comparison between soft constraints and hard constraints, GD^{2}NMF outperforms SGDNMF in most cases(Table 2). Moreover, we vary the label ratio from 0.1 to 0.4 and compare the performance (NMI) of GD^{2}NMF and SGDNMF. Results shown in Table R2 varify that soft constraint (GD^{2}NMF) has a better performance than hard constraint (SGDNMF).

Table R2: Comparison of GD^{2}NMF and SGDNMF on the CMUPIE dataset by variying the label ratio from 0.1 to 0.4

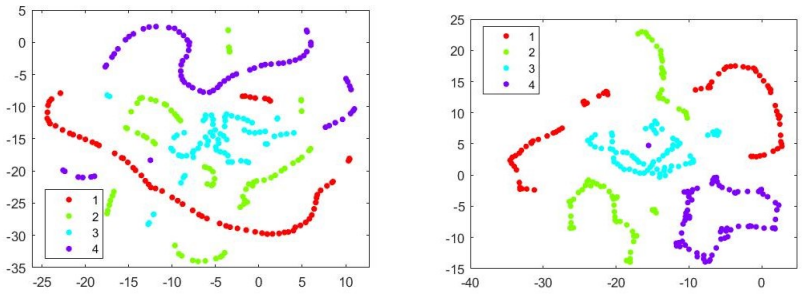
	0.1	0.2	0.3	0.4
GD^{2}NMF	82.83	83.99	84.98	84.64
SGDNMF	76.69	80.59	81.88	82.56

3.5 Visualization of Features on Different Layers

To evaluate the information obtained at different layers of GDNMF, we perform an experiment to visualize its layer-wise feature map on the COIL20 dataset. As shown in Fig.R3, GDNMF extracts hierarchical feature representations, which improves the discriminative ability layer by layer.

3.6 Case Study of Multi-Layer Structure

In the experiment, we choose to select a two-layer structure for GDNMF and GD^{2}NMF, which usually leads to a better result than other choices. To verify that, we conduct an experiment to compare the performance of GD^{2}NMF with different layers (α is set to 100 and decay rate r is set to 0.3), and results in Fig.R4 validate our assumption that a two-layer structure usually have the best performance. When the number of layers



(a) 1st layer (b) 2nd layer

Figure R3: Visualization of feature maps at different layers of GDNMF ($m = 2$) by t-SNE [Van der Maaten and Hinton, 2008]. Samples are better separated in the 2nd layer than in the 1st layer.

is greater than 2, its performance drops significantly, probably because the structure is too complex and easy to overfit.

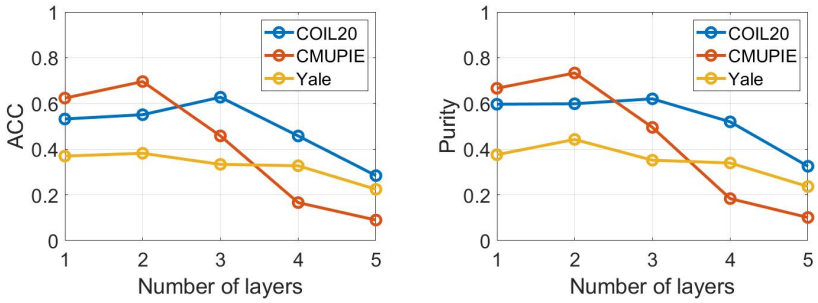


Figure R4: Performance evaluation of GD^{2}NMF on the CMUPIE, COIL20 and Yale datasets by varying the number of layers from 1 to 5.

References

[1] Meng, Yang, et al. "Semi-supervised graph regularized deep NMF with bi-orthogonal constraints for data representation." IEEE transactions on neural networks and learning systems 31.9 (2019): 3245-3258.

[2] Liu, Haifeng, et al. "Constrained nonnegative matrix factorization for image representation." IEEE Transactions on Pattern Analysis and Machine Intelligence 34.7 (2011): 1299-1311.

[3] Zhao, Yang, Huiyang Wang, and Jihong Pei. "Deep non-negative matrix factorization architecture based on underlying basis images learning." IEEE Transactions on Pattern Analysis and Machine Intelligence 43.6 (2019): 1897-1913.

[4] Trigeorgis, George, et al. ”A deep matrix factorization method for learning attribute representations.” IEEE transactions on pattern analysis and machine intelligence 39.3 (2016): 417-429.