report

Assignment 1

John Hu (zehu4485, 500395897)
Implemented NMF, L_1-NMF, $L_{2,1}$-NMF
Wrote most of report

Nicholas Grasevski (ngra5777, 500710654)
Implemented KL-NMF, tanh-NMF, CIM-NMF
Wrote experiment part of report

Tutor: Yu Yao

September 17, 2022

Abstract

In this report, we will explore the different NMF algorithms, including traditional NMF, L_1-NMF, $L_{2,1}$-NMF, KL-NMF, CIM-NMF and tanh-NMF. Through experimenting on ORL dataset and CroppedYaleB dataset with arbitrary induced noises, we found KL-NMF yields the best results on a clean dataset, whilst CIM-NMF is the most robust out of all algorithms.

1 Introduction

The Non-negative Matrix Factorization (NMF) is a powerful tool for analysing high dimensional data, e.g. images, movie ratings, stock market values. NMF can be said to be an approximation algorithm, where a higher dimensional matrix V is approximated by two lower rank matrixes W, H such that $V \approx WH$. [**wang-zhang**] Here the matrix W can be also considered as the weight of matrix H, hence the individual component of V is the weight sum of H, demonstrated as following:

$$V_i = \begin{bmatrix} w_{i,1} & w_{i,2} & \dots & w_{i,k-1} & w_{i,k} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \dots \\ h_{k-1} \\ h_k \end{bmatrix} \tag{1}$$

In this report, comparison different types of NMF algorithm will be performed. All of the algorithms will be used on two image datasets, and the result in terms of efficiency and accuracy of the algorithms will be shown and discussed.

2 Related work

2.1 Basic NMF and Multiplicative Update Rule (MUR)

The traditional or basic NMF has an objective function of $min_{W \leq 0, H \leq 0} \|V - WH\|_F^2$. [**lee-seung**] It was evident that the basic NMF's objective function is L_2-norm; it means the NMF can be sensitive to noise. The multiplicative update rule uses the fact that the Euclidean distance $\|V - WH\|$ is non-increasing under the rule:

$$H_{kn} \leftarrow H_{kn} \frac{\left(W^T V\right)_{kn}}{\left(W^T WH\right)_{kn}} \tag{2}$$

$$W_{mk} \leftarrow W_{mk} \frac{\left(VH^T\right)_{mk}}{\left(WHH^T\right)_{mk}} \tag{3}$$

Note that the Euclidean distance is invariant under these updates if and only if W and H are at a stationary point. [**marques-maciel-naviner-cai-yang**] The simplicity, elegance and computational tractability of the multiplicative update rule made it a popular choice for computing NMFs, even extending its use to semi-nonnegative matrix factorization where H is non-negative, but X can be both negative and non-negative. [**ding-li-jordan**]

2.2 L1 and L2 regularized NMF

The NMF uses a few crucial components of the data to construct an approximation, which means that NMF produces a sparse representation. [**luo-peng-fan**]

This is a handy feature of the NMF algorithm as most data are sparse in the transform domain, e.g. image in curvelet domain, frequency in Fourier transformation. The L_1 and L_2 regularized NMF tries to improve the basic NMF by implementing sparse coding and avoiding overfitting. The L_p norm can be shown as below:

$$\|a\|_p = \left(\sum_{j=1}^{k} |a|^p\right)^{\frac{1}{p}} \tag{4}$$

Hence the objective function of L_1 and L_2 regularized NMF is calculated by adding the sparsity, in this case the L_1 and L_2 norm:

$$min_{W \leq 0, H \leq 0} \|V - WH\|_F^2 + \lambda \Psi\left(R\right) \tag{5}$$

where $\Psi\left(R\right)$ is the sparsity constraint.

However, L_1 and L_2 norm based NMF can be sensitive to outliers and sometimes unstable. The regularization could improve stability but cannot eliminate the impact of outliers.

2.3 Correntropy Induced Matrix (CIM) NMF

Instead of the traditional Euclidean distance, CIM is utilising correntropy to calculate the similarity between two variables [**liu-pokharel-principe**]:

$$\hat{V_\sigma}(x, y) = \frac{1}{n}\sum_{i=1}^{n} k_\sigma (x_i - y_i) \tag{6}$$

where $k()$ is the kernel function, and Gaussian kernel is used. Hence the objective function is:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} 1 - \frac{1}{\sqrt{2\pi}\sigma}\exp\left(\frac{-(V - WH)_{ij}^2}{2\sigma^2}\right) \tag{7}$$

The mathematical property of Gaussian distribution enables the NMF to retain some robustness, but normalization should be considered when implementing this method to ensure the optimization is numerically stable. [**du-li-shen**]

2.4 Kullback-Leibler(KL)-NMF

The Kullback-Leibler divergence is another way that the similarity between two variables can be calculated. [**fevotte-idier**] In the famous Lee and Seung's NMF paper, a generalised KL divergence or IS-divergence is used. In this paper, we implement based on the KL-divergence. The distance or the similarity can be calculated as follows:

$$D_{\mathrm{KL}}(V|WH) = \sum_{ij} V_{ij}\log\left(\frac{V_{ij}}{(WH)_{ij}}\right) - V_{ij} + (WH)_{ij} \tag{8}$$

Hence the minimization of this function is the objective. This method could potentially improve robustness compared to traditional NMF. [**finesso-spreij**]

2.5 Half-Quadratic minimization

To compute and find the minimization of the some of the non-quadratic loss functions, the half quadratic minimization technique is used, more recently the half-quadratic programming algorithm is widely used. [**geman-chengda**] Half quadratic minimization uses half-quadratic theory [**nikolova-chan**] whereby adding an auxiliary variable can effectively minimize a non-quadratic or non-convex loss function. The technique can be applied to NMF, where the minimization can be solved by fixing the loss function to solve for the auxiliary variable, then use the auxiliary variable to update the loss function, then repeat this process until it converges. [**du-li-shen**]

2.6 Tanh NMF

Tanh NMF is a relatively new NMF algorithm which tries to build a robust NMF by utilising the property of the hyperbolic tangent function. [**shen-zhang-lan-liao-luo**] To use the tanh function for NMF purpose, Shen has parameterised the tanh function:

$$\tanh\left(ax^2\right) = \frac{e^{ax^2} - e^{-ax^2}}{e^{ax^2} + e^{-ax^2}} \tag{9}$$

Hence the similarity function can be written as:

$$\sum_{ij} \tanh\left(aE_{ij}^2\right) \tag{10}$$

where

$$E_{ij} = V_{ij} - \sum_{r}^{k} W_{ir} H_{rj} \tag{11}$$

Then by using the HQ algorithm:

$$\sum_{ij} \left[U_{ij} E_{ij}^2 + \Psi\left(U_{ij}\right)\right] \tag{12}$$

where U_{ij} is the auxiliary matrix, and $\Psi\left(U_{ij}\right)$ is the conjugate function of the loss function. It is worth noting that when E (the error term) is fixed, U can be proved to be the following through convex property and Fenchel-Moreau theorem [**shen-zhang-lan-liao-luo**]:

$$U_{ij} = a\left[1 - \tanh^2\left(a\left|E_{ij}\right|\right)\right] \tag{13}$$

This algorithm is a robust and novel method to perform NMF; however, the accuracy still needs improvement as we will show in later sections.

3 Methods

3.1 Pre-processing

We apply the following preprocessing to the images:

1. The image size is scaled down to reduce computation time
2. The image data is scaled down from the range $[0, 255]$ to $[0, 1]$ (by dividing by 255), in order to ensure numerical stability and speed up the convergence
3. After applying noise, the image data is clipped to the range $\left[10^{-7}, 1\right]$ to avoid division by zero in the multiplicative updates as well as ignoring impossible values (darker than pitch black, brighter than maximum brightness)

Algorithm	**Objective function**	$w\left(X,W,H\right)$
NMF	$\left\Vert V-WH\right\Vert_F^2$	1
KL-NMF	$\sum_{ij} V_{ij}\log\left(\frac{V_{ij}}{\left(WH\right)_{ij}}\right)-V_{ij}+\left(WH\right)_{ij}$	1
L_1-NMF	$\left\Vert V-WH\right\Vert_1$	$\frac{1}{\left\Vert V-WH\right\Vert_1}$
$L_{2,1}$-NMF	$\left\Vert V-WH\right\Vert_{2,1}$	$\frac{1}{\left\Vert V-WH\right\Vert_{2,1}}$
CIM-NMF	$\sum_{i=1}^m\sum_{j=1}^n 1-\frac{1}{\sqrt{2\pi}\sigma}e^{\frac{-(V-WH)_{ij}^2}{2\sigma^2}}$	$\exp\left(\frac{-E^2}{2\sigma^2}\right)$ where $\sigma^2=\frac{1}{2mn}\sum_{i=1}^m\sum_{j=1}^n E_{ij}^2$
tanh-NMF	$\sum_{ij}\tanh\left(aE_{ij}^2\right)$ where $a=\frac{nmp}{\sum_{i=1} m\sum_{j=1} nE_{ij}^2}$ and $p\in\{1\}$	$a\left[1-\tanh^2\left(a\left\vert E\right\vert\right)\right]$ where $a=\frac{nmp}{\sum_{i=1}^m\sum_{j=1}^n E_{ij}^2}$ and $p\in\{1\}$

Table 1: Algorithms and objective functions, where $E=V-WH$

Algorithm 1 Weighted NMF with β-divergence and regularization

```
Require: $V \in \mathbb{R}^{m\times n}, k \in \mathbb{Z}^+, \beta, \lambda_{W1}, \lambda_{W2}, \lambda_{H1}, \lambda_{H2} \in \mathbb{R}, w \in \left(\mathbb{R}^{m\times n}, \mathbb{R}^{m\times k}, \mathbb{R}^{k\times n}\right) \to \mathbb{R}^{m\times n}$
Ensure: $W \in \mathbb{R}^{m\times k}, H \in \mathbb{R}^{k\times n}$
  $W, H \leftarrow U\left(0,1\right), U\left(0,1\right)$
  repeat
    $U \leftarrow w\left(X,W,H\right)$
    $W \leftarrow W \otimes \frac{\left[U\otimes V\otimes (WH)^{\beta-2}\right]H^T}{\left[U\otimes (WH)^{\beta-1}\right]H^T+\lambda_{W1}+\lambda_{W2}W}$
    $H \leftarrow H \otimes \frac{W^T\left[U\otimes V\otimes (WH)^{\beta-2}\right]}{W^T\left[U\otimes (WH)^{\beta-1}\right]+\lambda_{H1}+\lambda_{H2}H}$
  until converged
```

3.2 Algorithms and objective functions

All the NMF algorithms used and their objective functions are shown in table 1. All algorithms use the weighted multiplicative update rule to compute W and H. Therefore iteratively, all algorithms can be implemented by adjusting weight according to its loss function. [**arora**] Algorithm 1 illustrates the generic procedure used by all of the NMF variants tested in this experiment.

Please note $\beta, \lambda_{W1}, \lambda_{W2}, \lambda_{H1}, \lambda_{H2}$ allows a broader range of extensions to the algorithm (different objective functions, different regularization) and combinations thereof than have been explored in this experiment. In this experiment, we set $\beta = 1$ for KL-NMF, and $\beta = 2$ for algorithms. No regularization is used, i.e. $\lambda_{W1} = \lambda_{W2} = \lambda_{H1} = \lambda_{H2} = 0$.

Before experimenting, the objective function shows that theoretically CIM and Tanh NMFs are more robust than other methods, as both Gaussian kernel and tanh function are not sensitive to outliers. It is due to comparing gaussian and tanh to a linear or quadratic function. Extreme values hava a relatively lesser impact when using gaussian or tanh function to update the weight matrix.

Each algorithm is applied to both datasets, and evaluation metrics are recorded.

3.3 Introduced noises

Noises are introduced into the data, and algorithms are evaluated again on the noisy data. The result of noisy data will be compared vertically and horizontally. It will allow us to understand the robustness of the algorithm.

3.3.1 Salt and pepper noise

During an analog-to-digital conversion, an image may obtain some dark pixels in bright regions and bright pixels in dark regions. [**noise**] Although modern techniques can effectively eliminate this noise, the salt and pepper noise was added to further examine the robustness of the algorithms. The noise was applied like so:

$$V = \hat{V} + 1\{P \wedge R\} - 1\{P \wedge \neg R\} \tag{14}$$

where V is the noisy image, $\hat{V}$ is the original image, $P \equiv z_1 \leq \sigma$, $R \equiv z_2 \leq r$, $z_1 \sim N(0, \sigma)$, $z_1 \sim N(0, r)$, $\sigma \in \{0.1, 0.2, 0.3, 0.4\}$ is the noise level hyperparameter and $r \in \{0.3\}$ is the white-to-black ratio hyperparameter. Figures 1 and 2 demonstrate how the various algorithms cope with this noise on the ORL and CroppedYaleB datasets respectively.

3.3.2 Laplace noise

Laplace noise refers to the noise in the statistical model used, as well as the noise during the processing of signals, for example, when taking a photo using a digital sensor, the sensor would have inherent noises due to its temperature, level of illumination and electrical wiring. Therefore, a perfect image with no noise is most impossible to obtain in practice, so when processing an image signal using NMF, we need to account for the

ORL-salt_and_pepper-3.png

Figure 1: ORL salt and pepper noise, $\sigma = 0.4$

CroppedYaleB-salt_and_pepper-3.png

Figure 2: CroppedYaleB salt and pepper noise, $\sigma = 0.4$

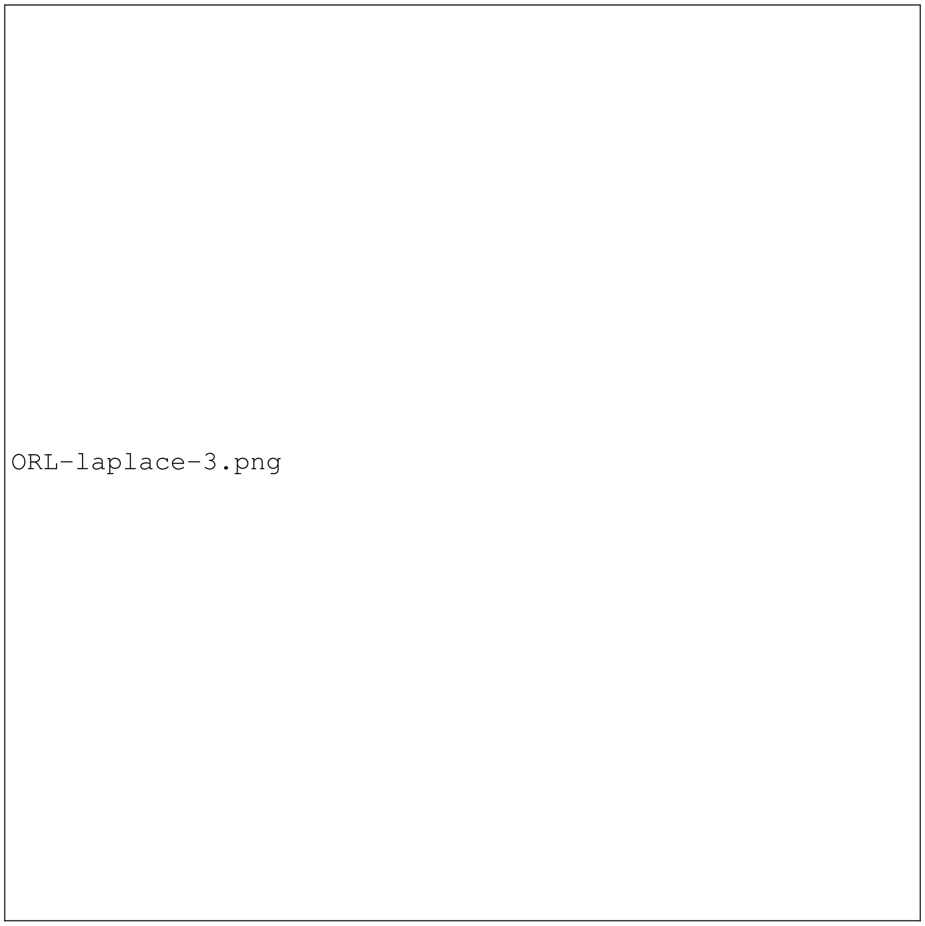

Figure 3: ORL laplace noise, $\sigma = 0.4$

CroppedYaleB-laplace-3.png

Figure 4: CroppedYaleB laplace noise, $\sigma = 0.4$

ORL-uniform-3.png

Figure 5: ORL uniform noise, $\sigma = 0.4$

noises. It is also worth to mention that noises are mostly random in nature. In principle the L_1-NMF should handle this type of noise well because the L_1 norm assumes a Laplace distribution. [**yang-zhang-yang-zhang**] The noise was applied like so:

$$V = \hat{V} + Z \tag{15}$$

where V is the noisy image, $\hat{V}$ is the original image, $Z \sim \text{Laplace}\,(0, \sigma)$ and $\sigma \in \{0.1, 0.2, 0.3, 0.4\}$ is the scale hyperparameter. Figures 3 and 4 demonstrate how the various algorithms cope with this noise on the ORL and CroppedYaleB datasets respectively.

CroppedYaleB-uniform-3.png

Figure 6: CroppedYaleB uniform noise, $\sigma = 0.4$

3.3.3 Uniform noise

Being digital, the pixels of a given image can sometimes be quantized to a number of discrete levels, for example if the original image had a low resolution. This is known as quantization noise [**noise**] We approximate this effect by adding uniform noise to the original image. The noise was applied like so:

$$V = \hat{V} + Z \tag{16}$$

where V is the noisy image, $\hat{V}$ is the original image, $Z \sim U(-\sigma, \sigma)$ and $\sigma \in \{0.1, 0.2, 0.3, 0.4\}$ is the noise level hyperparameter. Figures 5 and 6 demonstrates how the various algorithms cope with this noise on the ORL and CroppedYaleB datasets respectively.

3.3.4 Gaussian noise

Gaussian noise is noise with a normal distribution. It can arise in digital images due to sensor noise, poor illumination, high temperature or electronic circuit noise. [**gaussian**] Frobenius norm already assumes gaussian noise, thus standard NMF is well equipped to handle this kind of corruption. Nevertheless it is included for comparison as it is a common type of noise in images. The noise was applied like so:

$$V = \hat{V} + Z \tag{17}$$

where V is the noisy image, $\hat{V}$ is the original image, $Z \sim N\left(0, \sigma^2\right)$ and $\sigma \in \{0.1, 0.2, 0.3, 0.4\}$ is the standard deviation hyperparameter. Figures 7 and 8 demonstrate how the various algorithms cope with this noise on the ORL and CroppedYaleB datasets respectively.

4 Experiment

The experiment will evaluate the algorithms introduced in method on the ORL and CroppedYaleB datasets which will be explained in detail later. The algorithm will be evaluated in turn on each noise type listed above.

The algorithms are run on 5 different samplings of 90% of the data and the mean and standard deviation of the results are recorded. Due to the sensitivity of NMF to initialization, each algorithm is given the same (randomly chosen) initialization point in a given trial to ensure a fair comparison.

4.1 Dataset

The datasets used in this report are ORL, and CroppedYaleB. ORL is a facial image dataset, it contains 400 images with a size of $(112, 92)$. CroppedYaleB is a dataset cropped from YaleB, YaleB is a very large dataset with 5760 images, done by 10 subjects posing 9 photos under 64 different light conditions.

ORL-gaussian-3.png

Figure 7: ORL gaussian noise, $\sigma = 0.4$

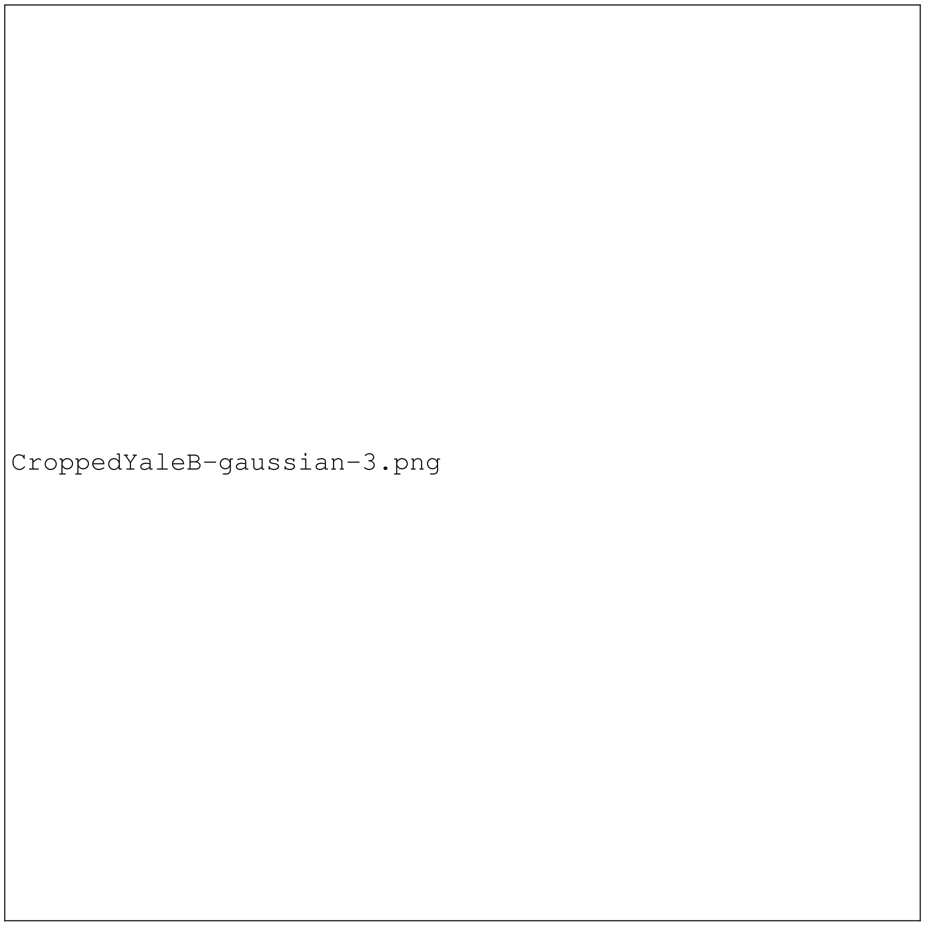

Figure 8: CroppedYaleB gaussian noise, $\sigma = 0.4$

4.2 Evaluation metrics

In order to perform a comprehensive comparison between the algorithms, several evaluation metrics are used: Relative Reconstruction Errors (RRE), Average Accuracy and Normalized Mutual Information (NMI). For the purposes of evaluation, we set the number of components to the number of classes (i.e. people) in the original datasets, so as to be able to meaningfully calculate the metrics described below.

4.2.1 Relative Reconstruction Errors

Relative Reconstruction Errors demonstrate how well the algorithm can regenerate the original data with or without added noise. It is calculated using the following formula:

$$\text{RRE} = \frac{\left\| \hat{V} - WH \right\|_F}{\left\| \hat{V} \right\|_F} \tag{18}$$

where $\hat{V}$ is the clean dataseet and W, H are the factorization of V.

4.2.2 Average Accuracy

The average accuracy is done according to K-means clustering, with K set to the number of classes in the dataset (number of individuals), then based on the clustering the average accuracy can be calculated using the prediction Y_{pred} as follows:

$$\text{Acc}\left(Y, Y_{\text{pred}}\right) = \frac{1}{n} \sum_{i=1}^{n} 1\left\{Y_{\text{pred}}\left(i\right) = Y\left(i\right)\right\} \tag{19}$$

4.2.3 Normalized Mutual Information

The normalized mutual information allows us to better understand the average accuracy, as it maximizes the effect of true prediction:

$$\text{NMI}\left(Y, Y_{\text{pred}}\right) = \frac{2I\left(Y, Y_{\text{pred}}\right)}{H\left(Y\right) + H\left(Y_{\text{pred}}\right)} \tag{20}$$

where $I\left(\cdot, \cdot\right)$ is mutual information and $H\left(\cdot\right)$ is entropy.

4.3 Results

Figures 9 and 10 show tanh-NMF and CIM-NMF performing well as the salt-and-pepper noise is increased. However for the other types of noise, they perform relatively poorly. Perhaps this is unsurprising, given that the other types of noise do not simulate outliers and furthermore they match too favorably to the underlying assumptions (NMF with Gaussian noise, L_1-NMF with Laplace noise, etc).

ORL-salt_and_pepper.png

Figure 9: ORL salt and pepper noise level vs evaluation metrics

CroppedYaleB-salt_and_pepper.png

Figure 10: CroppedYaleB salt and pepper noise level vs evaluation metrics

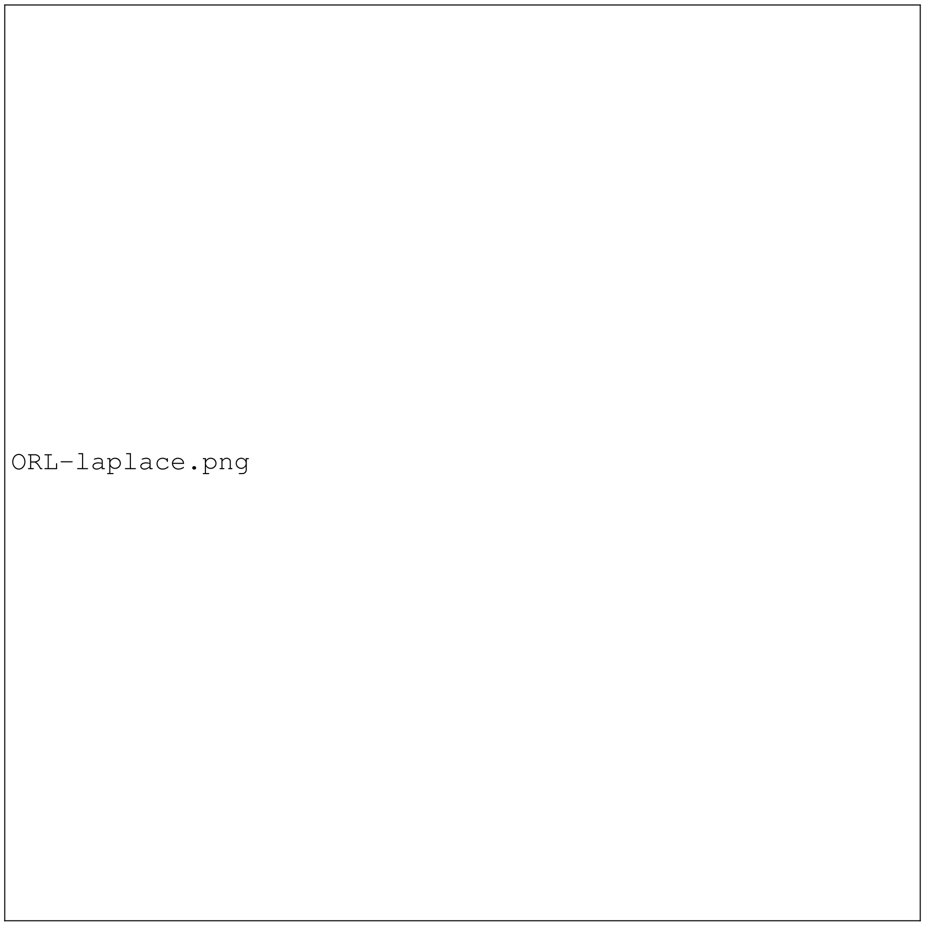

Figure 11: ORL laplace noise standard deviation vs evaluation metrics

CroppedYaleB-laplace.png

Figure 12: CroppedYaleB laplace noise standard deviation vs evaluation metrics

ORL-uniform.png

Figure 13: ORL uniform noise level vs evaluation metrics

CroppedYaleB-uniform.png

Figure 14: CroppedYaleB uniform noise level vs evaluation metrics

ORL-gaussian.png

Figure 15: ORL gaussian noise level vs evaluation metrics

CroppedYaleB-gaussian.png

Figure 16: CroppedYaleB gaussian noise level vs evaluation metrics

4.4 Discussion

From the result, we can observe that NMF, KL-NMF, L_1-NMF, and $L_{2,1}$-NMF have similar results, where the impact of noise on the accuracy, NMI and RRE are very much alike. These algorithms are all capable of generating good results without noise; however, they are susceptible to contaminated data because the result has shown that their accuracy decreases significantly when exposed to noise. The result matches the prediction made based on the mathematical property of the objective functions, cim-NMF or tanh-NMF are more robust than the other algorithms. Cim-NMF or tanh-NMF are less sensitive to noise, and they performed relatively well even with noise introduced.

However, there is a trade-off between robustness and accuracy. Although the robust algorithms are relatively insensitive to noise, a lower level of accuracy is shown in comparison to the other algorithms.

5 Conclusion

In this report, we have compared several different NMF algorithms. The best performing algorithm, on a clean dataset is the KL-NMF, the most robust algorithm being CIM-NMF. Based on the results, there is an apparent trade-off between robustness and accuracy. Perhaps the future research of NMF algorithm can focus on improving the accuracy while retaining the robust property of the algorithm.

A Running the code

- Dependencies: `pip install matplotlib scikit-learn`
- Usage: `./code/algorithm/algorithm.py`
- Help: `./code/algorithm/algorithm.py -h`

The full experiment takes approximately 3 hours to run on a 12-core Macbook Pro. The running time can be reduced by:

- reducing the number of trials: `-k 1`
- reducing the number of multiplicative updates: `-m 50`
- reducing the number of noise types: `-n salt_and_pepper laplace`
- reducing the number of algorithms: `-a nmf tanh_nmf`
- reducing the number of datasets: `-t ORL`

The runtime is roughly linearly proportional to each of the above factors. The results are printed to stdout incrementally and the script can be cancelled at any time.

B Detailed Experiment Results

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	19.31 ± 0.05	19.37 ± 0.04	19.31 ± 0.05	19.28 ± 0.05	16.88 ± 0.13	44.84 ± 0.62
0.2	24.99 ± 0.07	25.40 ± 0.09	24.99 ± 0.07	24.99 ± 0.07	16.29 ± 0.17	58.80 ± 0.33
0.3	30.17 ± 0.12	30.99 ± 0.11	30.17 ± 0.12	30.18 ± 0.12	18.33 ± 0.25	66.35 ± 0.47
0.4	35.04 ± 0.16	36.25 ± 0.16	35.04 ± 0.16	35.06 ± 0.18	26.50 ± 0.36	74.83 ± 0.51

Table 2: RRE(%) on ORL dataset with salt-and-pepper noise (mean ± std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	64.72 ± 3.49	67.00 ± 1.78	64.72 ± 3.49	66.61 ± 1.72	71.06 ± 1.68	53.89 ± 2.31
0.2	47.44 ± 2.30	49.17 ± 5.15	47.44 ± 2.30	48.39 ± 2.51	71.72 ± 3.14	64.50 ± 3.38
0.3	32.89 ± 0.97	32.50 ± 0.98	32.89 ± 0.97	32.72 ± 1.68	68.94 ± 2.46	62.83 ± 1.91
0.4	25.28 ± 0.91	25.50 ± 1.83	25.28 ± 0.91	25.00 ± 1.18	55.94 ± 3.73	38.44 ± 1.96

Table 3: Acc(%) on ORL dataset with salt-and-pepper noise (mean ± std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	78.66 ± 2.70	81.06 ± 1.26	78.66 ± 2.70	79.74 ± 1.02	84.34 ± 1.34	70.59 ± 1.79
0.2	64.58 ± 1.72	65.42 ± 4.01	64.58 ± 1.72	65.35 ± 1.43	84.24 ± 1.09	80.40 ± 1.73
0.3	51.66 ± 0.31	51.43 ± 0.69	51.66 ± 0.31	52.26 ± 1.64	82.88 ± 1.88	79.30 ± 1.45
0.4	44.30 ± 1.56	44.29 ± 1.88	44.30 ± 1.56	44.36 ± 1.00	72.83 ± 2.03	59.08 ± 1.38

Table 4: NMI(%) on ORL dataset with salt-and-pepper noise (mean ± std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	15.07 ± 1.67	14.92 ± 1.59	15.07 ± 1.67	15.13 ± 1.65	22.54 ± 1.33	37.25 ± 1.86
0.2	17.76 ± 4.93	17.61 ± 4.87	17.76 ± 4.93	17.82 ± 4.92	24.69 ± 4.10	37.01 ± 2.68
0.3	21.15 ± 8.61	21.01 ± 8.56	21.15 ± 8.61	21.20 ± 8.59	27.53 ± 7.49	37.81 ± 3.45
0.4	24.91 ± 12.33	24.79 ± 12.29	24.91 ± 12.33	24.96 ± 12.32	30.81 ± 11.12	39.41 ± 5.49

Table 5: RRE(%) on ORL dataset with laplace noise (mean ± std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	73.00 ± 1.76	75.50 ± 3.78	73.00 ± 1.76	75.28 ± 3.54	61.33 ± 1.90	29.67 ± 2.13
0.2	72.44 ± 2.02	76.11 ± 1.44	72.44 ± 2.02	74.50 ± 3.35	63.00 ± 2.57	28.67 ± 3.63
0.3	73.39 ± 0.90	75.50 ± 1.41	73.39 ± 0.90	76.50 ± 2.28	61.00 ± 4.60	28.33 ± 3.93
0.4	73.06 ± 1.98	74.33 ± 2.18	73.06 ± 1.98	75.56 ± 3.29	60.94 ± 4.63	27.94 ± 6.52

Table 6: Acc(%) on ORL dataset with laplace noise (mean ± std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	85.75 ± 0.80	87.02 ± 1.72	85.75 ± 0.80	85.59 ± 1.84	78.34 ± 1.48	47.25 ± 2.40
0.2	84.78 ± 1.28	87.13 ± 0.98	84.78 ± 1.28	85.66 ± 1.91	78.57 ± 2.13	47.35 ± 2.62
0.3	85.32 ± 0.44	86.35 ± 0.62	85.32 ± 0.44	87.53 ± 1.30	77.31 ± 3.45	46.35 ± 4.24
0.4	85.47 ± 0.62	86.21 ± 1.11	85.47 ± 0.62	86.09 ± 2.08	77.36 ± 3.56	45.85 ± 6.61

Table 7: NMI(%) on ORL dataset with laplace noise (mean ± std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	13.96 ± 0.04	13.90 ± 0.05	13.96 ± 0.04	14.01 ± 0.02	21.41 ± 0.07	37.97 ± 0.24
0.2	14.35 ± 0.04	14.35 ± 0.05	14.35 ± 0.04	14.39 ± 0.02	21.28 ± 0.09	36.70 ± 0.25
0.3	15.04 ± 0.05	15.13 ± 0.06	15.04 ± 0.05	15.07 ± 0.03	21.96 ± 0.13	35.15 ± 0.21
0.4	16.04 ± 0.06	16.24 ± 0.08	16.04 ± 0.06	16.07 ± 0.04	23.60 ± 0.11	34.20 ± 0.06

Table 8: RRE(%) on ORL dataset with uniform noise (mean ± std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	72.83 ± 4.05	76.06 ± 1.12	72.83 ± 4.05	74.00 ± 3.80	61.83 ± 2.27	23.78 ± 1.52
0.2	73.00 ± 2.92	73.44 ± 2.54	73.00 ± 2.92	72.61 ± 3.42	60.50 ± 2.04	21.44 ± 0.48
0.3	73.00 ± 2.29	73.28 ± 2.35	73.00 ± 2.29	73.33 ± 2.13	53.83 ± 2.20	19.11 ± 0.73
0.4	72.78 ± 2.73	71.94 ± 2.72	72.78 ± 2.73	70.50 ± 2.04	44.39 ± 3.23	18.89 ± 0.53

Table 9: Acc(%) on ORL dataset with uniform noise (mean ± std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	85.08 ± 1.97	86.82 ± 0.93	85.08 ± 1.97	85.84 ± 1.69	78.80 ± 0.79	42.28 ± 2.17
0.2	85.11 ± 1.91	85.56 ± 0.70	85.11 ± 1.91	85.06 ± 1.80	76.61 ± 1.69	39.46 ± 0.77
0.3	84.90 ± 1.41	85.32 ± 1.38	84.90 ± 1.41	85.01 ± 1.39	70.81 ± 1.55	38.14 ± 1.33
0.4	84.13 ± 1.27	83.75 ± 1.03	84.13 ± 1.27	83.64 ± 1.32	63.26 ± 1.57	37.00 ± 1.51

Table 10: NMI(%) on ORL dataset with uniform noise (mean ± std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	34.26 ± 11.90	34.03 ± 11.93	34.26 ± 11.90	34.29 ± 11.89	39.17 ± 10.63	39.18 ± 4.29
0.2	60.56 ± 24.41	60.38 ± 24.48	60.56 ± 24.41	60.58 ± 24.41	64.86 ± 23.76	59.78 ± 18.95
0.3	81.61 ± 31.78	81.49 ± 31.87	81.61 ± 31.78	81.63 ± 31.78	86.19 ± 31.88	80.27 ± 29.36
0.4	94.70 ± 33.17	94.64 ± 33.28	94.70 ± 33.17	94.73 ± 33.18	97.62 ± 32.15	93.44 ± 32.70

Table 11: RRE(%) on ORL dataset with gaussian noise (mean ± std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	73.00 ± 2.39	74.33 ± 2.51	73.00 ± 2.39	74.39 ± 1.59	59.39 ± 2.92	22.56 ± 2.29
0.2	71.78 ± 1.74	74.83 ± 3.02	71.78 ± 1.74	73.11 ± 1.84	48.39 ± 8.46	19.72 ± 2.73
0.3	59.89 ± 12.59	62.22 ± 13.11	59.89 ± 12.59	61.06 ± 12.75	35.94 ± 19.12	19.17 ± 2.10
0.4	42.00 ± 24.22	42.00 ± 24.31	42.00 ± 24.22	40.83 ± 24.13	30.83 ± 16.48	17.89 ± 1.67

Table 12: Acc(%) on ORL dataset with gaussian noise (mean $\pm$ std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	85.50 ± 1.33	85.59 ± 1.06	85.50 ± 1.33	85.65 ± 1.11	75.76 ± 1.55	41.33 ± 3.34
0.2	84.56 ± 0.84	86.30 ± 1.85	84.56 ± 0.84	84.95 ± 0.92	66.70 ± 7.21	37.15 ± 3.43
0.3	75.32 ± 9.81	77.36 ± 10.60	75.32 ± 9.81	75.99 ± 10.05	53.08 ± 17.38	37.54 ± 1.69
0.4	58.32 ± 21.58	58.10 ± 22.34	58.32 ± 21.58	56.90 ± 22.04	47.68 ± 17.00	35.47 ± 2.14

Table 13: NMI(%) on ORL dataset with gaussian noise (mean $\pm$ std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	21.88 ± 0.07	22.62 ± 0.07	21.88 ± 0.07	21.90 ± 0.08	24.54 ± 0.23	57.43 ± 0.36
0.2	26.22 ± 0.04	27.33 ± 0.07	26.22 ± 0.04	26.25 ± 0.04	24.67 ± 0.20	66.75 ± 0.40
0.3	31.82 ± 0.08	32.79 ± 0.10	31.82 ± 0.08	31.84 ± 0.08	27.40 ± 0.25	74.57 ± 0.15
0.4	37.88 ± 0.10	38.53 ± 0.12	37.88 ± 0.10	37.91 ± 0.11	34.20 ± 0.38	81.09 ± 0.16

Table 14: RRE(%) on CroppedYaleB dataset with salt-and-pepper noise (mean $\pm$ std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	21.97 ± 0.99	20.95 ± 1.03	21.97 ± 0.99	21.08 ± 1.51	19.35 ± 0.51	10.59 ± 0.18
0.2	18.90 ± 1.21	17.28 ± 1.30	18.90 ± 1.21	19.52 ± 1.05	20.44 ± 0.63	13.02 ± 0.52
0.3	16.68 ± 0.77	14.44 ± 0.78	16.68 ± 0.77	16.22 ± 0.79	20.35 ± 0.61	14.93 ± 0.71
0.4	12.80 ± 0.55	12.09 ± 0.32	12.80 ± 0.55	13.15 ± 0.56	18.77 ± 0.46	12.54 ± 0.90

Table 15: Acc(%) on CroppedYaleB dataset with salt-and-pepper noise (mean $\pm$ std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	29.25 ± 1.14	29.10 ± 1.41	29.25 ± 1.14	28.93 ± 1.59	23.63 ± 1.92	12.40 ± 0.79
0.2	26.07 ± 0.96	24.84 ± 1.37	26.07 ± 0.96	25.69 ± 0.89	25.67 ± 0.75	16.69 ± 1.00
0.3	21.81 ± 0.67	19.72 ± 1.62	21.81 ± 0.67	20.95 ± 0.67	26.72 ± 1.08	18.41 ± 0.51
0.4	16.03 ± 0.70	14.66 ± 0.90	16.03 ± 0.70	16.08 ± 0.50	24.06 ± 0.92	15.15 ± 0.88

Table 16: NMI(%) on CroppedYaleB dataset with salt-and-pepper noise (mean $\pm$ std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	20.89 ± 2.00	21.12 ± 1.76	20.89 ± 2.00	21.08 ± 1.97	29.41 ± 1.26	52.33 ± 0.55
0.2	24.18 ± 6.16	24.37 ± 5.88	24.18 ± 6.16	24.35 ± 6.12	31.82 ± 4.48	53.20 ± 1.39
0.3	28.32 ± 10.93	28.48 ± 10.69	28.32 ± 10.93	28.48 ± 10.90	35.20 ± 8.73	54.99 ± 3.43
0.4	32.94 ± 15.83	33.07 ± 15.62	32.94 ± 15.83	33.07 ± 15.80	39.16 ± 13.37	57.26 ± 6.27

Table 17: RRE(%) on CroppedYaleB dataset with laplace noise (mean ± std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	23.32 ± 0.64	23.85 ± 0.97	23.32 ± 0.64	24.07 ± 1.74	13.93 ± 0.69	8.56 ± 0.30
0.2	23.21 ± 0.76	24.48 ± 1.20	23.21 ± 0.76	23.40 ± 1.03	13.95 ± 1.06	8.69 ± 0.17
0.3	23.52 ± 0.83	23.77 ± 1.23	23.52 ± 0.83	22.54 ± 0.73	13.44 ± 1.44	8.62 ± 0.15
0.4	22.04 ± 1.80	23.96 ± 1.29	22.04 ± 1.80	22.85 ± 1.34	13.16 ± 1.49	8.40 ± 0.31

Table 18: Acc(%) on CroppedYaleB dataset with laplace noise (mean ± std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	31.57 ± 0.83	31.39 ± 1.61	31.57 ± 0.83	31.84 ± 1.56	16.67 ± 1.00	8.93 ± 0.46
0.2	31.28 ± 0.74	32.45 ± 1.92	31.28 ± 0.74	32.10 ± 2.72	16.60 ± 2.25	9.30 ± 0.24
0.3	32.01 ± 1.41	31.67 ± 1.66	32.01 ± 1.41	31.74 ± 1.42	15.86 ± 2.52	9.40 ± 0.42
0.4	29.96 ± 1.54	30.78 ± 2.32	29.96 ± 1.54	30.59 ± 2.17	15.55 ± 2.37	8.69 ± 0.74

Table 19: NMI(%) on CroppedYaleB dataset with laplace noise (mean ± std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	19.47 ± 0.07	19.86 ± 0.08	19.47 ± 0.07	19.65 ± 0.12	28.52 ± 0.23	51.79 ± 0.92
0.2	19.68 ± 0.08	20.02 ± 0.06	19.68 ± 0.08	19.81 ± 0.10	28.20 ± 0.23	53.12 ± 0.90
0.3	20.14 ± 0.09	20.42 ± 0.06	20.14 ± 0.09	20.21 ± 0.09	28.24 ± 0.18	56.11 ± 0.88
0.4	20.85 ± 0.09	21.11 ± 0.06	20.85 ± 0.09	20.88 ± 0.07	28.77 ± 0.15	58.10 ± 1.16

Table 20: RRE(%) on CroppedYaleB dataset with uniform noise (mean ± std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	23.40 ± 1.08	24.01 ± 1.05	23.40 ± 1.08	23.21 ± 0.96	13.65 ± 0.48	8.59 ± 0.14
0.2	22.86 ± 1.06	23.72 ± 0.79	22.86 ± 1.06	23.06 ± 1.20	12.97 ± 0.36	8.34 ± 0.16
0.3	22.50 ± 0.72	23.70 ± 1.71	22.50 ± 0.72	22.31 ± 0.55	12.22 ± 0.63	8.22 ± 0.08
0.4	22.80 ± 0.77	22.87 ± 0.92	22.80 ± 0.77	21.99 ± 1.12	11.27 ± 0.32	8.00 ± 0.21

Table 21: Acc(%) on CroppedYaleB dataset with uniform noise (mean ± std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	32.03 ± 0.96	31.99 ± 1.08	32.03 ± 0.96	31.09 ± 1.21	16.77 ± 0.72	9.05 ± 0.50
0.2	30.56 ± 0.40	31.92 ± 0.72	30.56 ± 0.40	31.34 ± 0.97	15.67 ± 0.89	8.82 ± 0.24
0.3	31.56 ± 1.74	31.30 ± 1.08	31.56 ± 1.74	30.77 ± 0.81	14.75 ± 0.84	9.18 ± 0.69
0.4	31.39 ± 1.11	29.92 ± 1.04	31.39 ± 1.11	30.19 ± 1.59	12.82 ± 0.31	9.16 ± 0.14

Table 22: NMI(%) on CroppedYaleB dataset with uniform noise (mean ± std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	45.13 ± 15.14	44.87 ± 15.11	45.13 ± 15.14	45.26 ± 15.10	48.73 ± 12.92	59.72 ± 5.98
0.2	80.87 ± 33.04	80.65 ± 33.12	80.87 ± 33.04	80.93 ± 33.03	82.34 ± 31.01	84.27 ± 22.48
0.3	114.11 ± 47.41	113.97 ± 47.53	114.11 ± 47.41	114.18 ± 47.40	114.82 ± 45.87	113.24 ± 39.04
0.4	141.66 ± 56.84	141.58 ± 56.97	141.66 ± 56.84	141.75 ± 56.87	142.30 ± 56.05	140.89 ± 52.29

Table 23: RRE(%) on CroppedYaleB dataset with gaussian noise (mean ± std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	21.66 ± 0.76	22.48 ± 0.83	21.66 ± 0.76	21.36 ± 1.37	11.55 ± 0.82	8.21 ± 0.25
0.2	19.47 ± 2.42	19.41 ± 2.53	19.47 ± 2.42	19.45 ± 3.13	10.08 ± 1.58	7.97 ± 0.23
0.3	15.59 ± 4.60	16.05 ± 4.90	15.59 ± 4.60	15.49 ± 4.93	9.22 ± 1.26	7.84 ± 0.38
0.4	13.05 ± 5.93	13.15 ± 5.77	13.05 ± 5.93	12.63 ± 5.41	8.81 ± 1.23	7.85 ± 0.32

Table 24: Acc(%) on CroppedYaleB dataset with gaussian noise (mean ± std)

σ	nmf	kl-nmf	l1-nmf	l21-nmf	cim-nmf	tanh-nmf
0.1	30.13 ± 1.14	30.73 ± 1.89	30.13 ± 1.14	30.49 ± 2.00	12.96 ± 1.55	8.69 ± 0.37
0.2	26.28 ± 3.37	26.07 ± 3.27	26.28 ± 3.37	26.50 ± 4.40	10.66 ± 2.58	8.18 ± 0.40
0.3	19.69 ± 8.48	19.67 ± 7.83	19.69 ± 8.48	19.85 ± 8.24	9.58 ± 2.85	7.61 ± 1.03
0.4	15.37 ± 9.62	15.21 ± 8.78	15.37 ± 9.62	14.38 ± 8.90	9.32 ± 2.55	7.48 ± 0.70

Table 25: NMI(%) on CroppedYaleB dataset with gaussian noise (mean ± std)