Analysis of the NMF algorithms in Facial Recognition

TRINH Van Hoan VUONG Tung Duong
Department of Mathematics
Hong Kong University of Science and Technology
{vhtrinh,tdvuong}@connect.ust.hk

Abstract

We examine the Non-negative Matrix Factorization techniques and its applications in Facial Recognition. Firstly, we represent the NMF algorithms using the L_2-norm and the KL-divergence as the objective function, then examine the convergence property of NMF with KL-divergence and propose theoretical ideas to guarantee this property. Secondly, we analyze both the NMF algorithms' performance in Facial Recognition using a series of experiments on face datasets with different metrics and different simulated noise scenarios.

1 Introduction

Non-negative matrix factorization is used as a tool for data reduction similar to SVD, PCA or VQ, but only allows additive combination of the bases to construct the data by the non-negativity constraint[1]. The problem can be stated as follows: Given $V \in \mathbb{R}^{n\times m}$ with non-negative entries. Find $W \in \mathbb{R}^{n\times r}$, $H \in \mathbb{R}^{r\times m}$ such that

$$V \approx WH$$

and entries of W, H must be non-negative. The objective function can be chosen to be the L_2-norm

$$F(W,H) = ||V - WH||_2$$

or other metrics such as the proposed Kullback-Leibler (KL) divergence in [1]

$$F(W,H) = \sum_{i=1}^{n}\sum_{j=1}^{m}(V_{ij}\log\frac{V_{ij}}{(WH)_{ij}} - V_{ij} + (WH)_{ij})$$

One common technique to solve this problem is Multiplicative Update Rule (MUR), which has different forms for different choice of the objective function .

2 Related Work

2.1 NMF with the L_2-norm

If the L_2-norm is chosen as the objective function. D. Lee and H. Seung[1] have proven that the following update rules:

$$H_{ab} \leftarrow H_{ab}\frac{(W^TV)_{ab}}{(W^TWH)_{ab}}$$
$$W_{ia} \leftarrow W_{ia}\frac{(VH^T)_{ia}}{(WHH^T)_{ia}}$$

can be used to minimize the objective function.

3 NMF with the Kullback-Leibler Divergence

D. Lee and H. Seung[1] have proved that the multiplicative update ensure the convergence of the objective function. However, it doesn't mean W and H converge as well. One more problem is that the convergence point might not a stationary point.

In this section, we propose several modifications to ensure theoretical results for the KL-divergence algorithm, which is that W and H have a limit point and the limit point satisfies a necessary condition: the partial gradient is 0 when the limit point is non-zero.

3.1 Multiplicative Update Rule

Since the problem is symmetric for W and H, it is sufficient to derive the update rules for H only.

From the objective function, we can compute the partial gradient

$$\frac{\partial F}{\partial H_{ab}} = -\sum_{i=1}^{n} W_{ia} \frac{V_{ib}}{(WH)_{ib}} + \sum_{i=1}^{n} W_{ia}$$

$$\frac{\partial^2 F}{\partial H_{ab}^2} = \sum_{i=1}^{n} W_{ia}^2 \frac{V_{ib}}{((WH)_{ib})^2} \geq 0$$

Hence F is convex w.r.t each H's entry. In order to ensure the non-negativity, D. Lee and H. Seung[1] have chosen the appropriate step size such that the update can be reformed into the multiplicative update.

$$\eta_{ab} = \frac{H_{ab}}{\sum_{i=1}^{n} W_{ia}}$$

With this step size, the gradient descent is equivalent to:

$$H_{ab} \leftarrow H_{ab} \frac{\sum_{i=1}^{n} W_{ia} V_{ib}}{(WH)_{ib} \sum_{k=1}^{n} W_{ka}}$$

3.1.1 Discussion about the convergence

First we will briefly explain why the objective function converges under the multiplicative update based on [2].

With W fixed, rewrite the objective function in term of sum of functions of each column h_j

$$F(W, H) = \sum_{j} f(h_j)$$

where

$$f(h_j) = \sum_{i=1}^{n} (v_i \log \frac{v_i}{(Wh)_i} - v_i + (Wh)_i)$$

and $v_i = V_{ij}$.

Define the auxiliary function $G(h, h')$ for $f(h)$ such that:

$$G(h, h') \geq F(h), G(h, h) = F(h)$$

The update

$$h^{t+1} = \arg\min_{h} G(h, h^t)$$

gives us an non-increasing sequence

$$F(h^0) \geq F(h^1) \geq ...$$

which converges to $F*$ since F is bounded below.

The appropriate auxiliary function is:

$$G(h, h^t) = \sum_{i} (v_i \log v_i - v_i + W_i h) - \sum_{i,a} v_i \alpha_a \log \frac{W_{ia} h_a}{\alpha_{ia}}$$

where α_{ia} is a function of h^t satisfy:

1. $\sum_a \alpha_{ia} = 1$
2. $\frac{W_{ia}ha}{\alpha_{ia}} = W_i h$ when $h^t = h$

Choosing $\alpha_a = \frac{W_{ia}h_a^t}{W_i h^t}$ meets the above condition.

This function is auxiliary function of $F_W(h)$ because:

$$G(h, h^t) \geq \sum_i (v_i \log v_i - v_i + W_i h) - \sum_i v_i \log(\sum_a W_{ia} h_a) = f(h)$$

and

$$G(h, h) = \sum_i (v_i \log v_i - v_i + W_i h) - \sum_{i,a} v_i \alpha_{ia} \log \frac{W_{ia} h_a}{\alpha_{ia}} = f(h)$$

since $\frac{W_{ia}ha}{\alpha_{ia}} = W_i h$

Using the partial gradient

$$\frac{\partial G(h, h^t)}{\partial h_a} = \sum_i (W_{ia} - v_i \alpha_{ia} \frac{1}{h_a})$$

we find the corresponding update, which is indeed the multiplicative update:

$$h_a^{t+1} = \frac{\sum_i v_i \alpha_{ia}}{\sum_i W_{ia}}$$

We propose some of the modification that will ensure that the algorithm is well-defined.

Specifically, the algorithm is not well-defined when:

1. If $\sum_i W_{ia} = 0$, then $W_{ia} = 0 \forall i$, the update is not well-defined and $G(h, h^t)$ has no lower bound as well. However, in this case, $f(h)$ is a constant function w.r.t. h_a and $\nabla_{h_a} f(h) = 0$, thus no update needed since we can consider this entry as converged to a stationary point.
2. If $W_i h^t = 0$ for some i. If $v_i \neq 0$, then $f(h^t) = \infty$, updating h_a^t to a constant positive number would improve it in the long run, since $W_i h$ increases after W_{ia} is updated as well, while still ensure the non-increasing property. If $v_i = W_i h^t = 0$, replace $W_i a - v_i \alpha_{ia} \frac{1}{h_a}$ by 0, since it is not a part of the objective function as well.

With these rules, the algorithm is well-defined.

As mentioned before, the algorithm doesn't ensure the convergence of each entry. We propose normalizing W, H after each update to ensure that W and H can have a limit point in the next section.

3.2 Normalization ensures a limit point

W and H are normalized as follows:

1. Update:
$$h_a^{t+1,u} = \frac{\sum_i v_i \alpha_{ia}}{\sum_i W_{ia}}$$
Similarly for $W^{t+1,u}$. (u here means "unnormalized", not index)
2. Normalize $H^{t+1,u}, W^{t+1,u}$ to H^{t+1}, W^{t+1} by W's column, i.e. i-th column of W has sum of 1 or 0.

Under this normalization, W's entries are bounded, hence they have a limit point by Bolzano-Weierstrass Theorem. We prove that H's entries are bounded too.

Indeed, if $h_a = H_{aj}$ is not bounded, there exists an increasing sub-sequence of h_a that tends to infinity:

$$\lim h_a^{(n_j)} = \infty$$

Consider $\{W_{ia}^{(n_j)}\}_j$, there exists a sub-sequence of that converges to W_{ia}^*. Argue similarly for all i, then there exists a sub-sequence m_j that $W_{ia}^{(m_j)}$ converges to W_{ia}^* for all i.

Notice that with respect to h_a, $f(h)$ contains $(Wh)_i - v_i \log(Wh)_i$ which goes to infinity if $(Wh)_i \geq W_{ia}h_a$ goes to infinity, hence $\lim_{j\to\infty} W_{ia}^{m_j} h_a^{m_j} < \infty$, hence $\lim_{j\to\infty} W_{ia}^{m_j} = 0$.

Therefore

$$\lim_{j\to\infty} \sum_i W_{ia}^{m_j} = 0$$

However, this column is normalized, thus:

$$\sum_i W_{ia}^{m_j} = 0$$

for j large enough.

As discussed before, under the situation $\sum_i W_{ia}^{m_j} = 0$, h_a is not updated, hence cannot go to infinity, which is a contradiction.

Therefore, h_a is bounded. Thus, H has a limit point $H_{aj}^* = h_a^*$.

3.3 Necessary condition for stationary convergence point

One of the necessary condition for this limit point to be a stationary point is that if the value of the limit entry is not zero, then the partial gradient w.r.t. that entry must be 0. We will prove that this condition is met under this modified algorithm.

First of all, we prove the following:

Lemma: If $h_a^* = H_{aj}^* \neq 0$, then $\lim_j h_a^{m_j,u} = h_a^*$.

Indeed, suppose for any $\epsilon > 0$, there exists a sub-sequence $\{m_j'\}$ of $\{m_j\}$ such that $|h_a^{m_j',u} - h_a^*| > \epsilon$.

Note that the objective function converges, hence:

$$\lim_j |F(W^{m_j'}, h_a^{m_j',u}) - F(W^{m_j'}, h_a^{m_j'})| = 0$$

Take the limit, notice that $\lim_j F(W^{m_j'}, h^{m_j'}) = F^* = F(W^*, H^*)$

$$\lim_j |F(W^*, h_a^{m_j',u}) - F(W^*, H^*)| = 0$$

Note that h_a^* is the minimum point of $F(W^*, h_a^*)$ where $F(W^*, h)$ is a convex function w.r.t. h_a.

We consider 2 cases:

1. If $h < h_a^* - \epsilon$ then
$$F(W^*, h) > F(W^*, h_a^* - \epsilon)$$
2. If $h > h_a^* + \epsilon$ then
$$F(W^*, h) > F(W^*, h_a^* + \epsilon)$$

Therefore

$$|F(W^*, h^{m_j',u}) - F^*| > \min\{|F(W^*, h_a^* + \epsilon) - F^*|, |F(W^*, h_a^* - \epsilon) - F^*|\} > 0$$

which is a positive constant with h_a^*, contradict with the fact that

$$\lim_j |F(W^*, h_a^{m_j',u}) - F(W^*, H^*)| = 0$$

Therefore the lemma is proven.

Recall the update in terms of gradient descent:

$$h_a^{m_j,u} = h_a^{m_j} - \nabla_{h_a} F(W^{m_j}, h^{m_j}) \times \frac{h_a^{m_j}}{\sum_i W_{ia}}$$

Take the limit of both side, where $j \to \infty$:

$$h_a^* = h_a^* - \nabla_{h_a} F(W^*, h^*) \times \frac{h_a^*}{\sum_i W_{ia}}$$

If $h_a^* \neq 0$, then $\nabla_{h_a} F(W^*, h^*) = 0$.

Hence, the necessary condition is achieved and therefore, the limit point is a stationary point.

4 Experimental Setup

After examining the NMF algorithms in a theoretical perspective, in the 2nd part of this study, we conducted a series of experiments to analyze the NMF algorithm with the KL-divergence and the NMF algorithm with the standard L_2-norm. Additionally, we compare the performance of these NMF algorithms using different metrics under a range of simulated noise scenarios to analyze their reactions in the presence of noise. The experiments are conducted using publicly available face datasets, the Yale face dataset and the ORL face dataset

4.1 SVD-based initialization

As the Multiplicative Update Rule is also an iterative method, It is very sensitive to the initialization of W and H. To avoid using the standard randormization, NNDSVD Method is introduced by C. Boutsidis and E.Gallopoulos in 2007 as an alternative SVD-based initialization[3].

Non-negative Double Singular Value Decomposition (NNDSVD) initiation method does not contain randomization, additionally, it is based on approximations of positive sections of the partial singular value factors of the data matrices, using an algebraic property of unit rank matrices. The NNDSVD technique has been proven to significantly enhance the non-negative matrix factorization (NMF)[3]. Therefore, we also implemented this initialization method and used it to aid NMF algorithms.

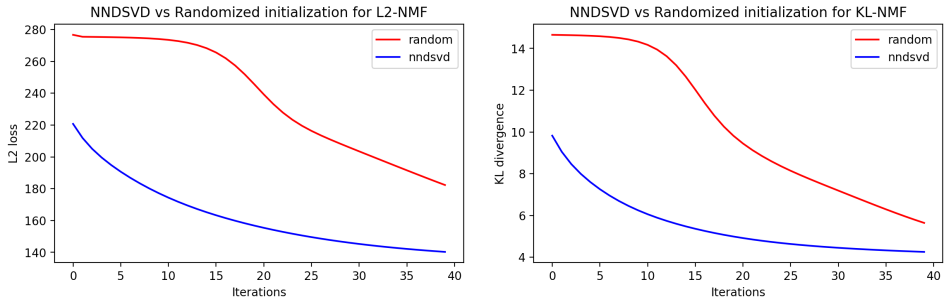

Figure 1: Effects of initialization methods on L_2-norm and KL divergence on the Yale face dataset

As can be seen in Figure1, applying NNSVD initialization provides better loss curve for the both NMF algorithms in comparison to the standard randomization.

4.2 Noise

In the real-world applications, the data are usually corrupted with noise. We present the formulation of the different types of noise implemented for analyzing the influence of noise to the NMF algorithms.

4.2.1 Salt and Pepper Noise

This is commonly referred to as Impulse Valued Noise or Data Drop Noise, is usually generated by malfunctioning of pixel elements in camera sensors or errors in conversion process. The Salt and

Pepper Noise corrupts some pixel values in the image, commonly by the maximum and minimum pixel values, 255 (Salt noise) and 0 (Pepper noise)[4].

The noise is generated by randomly changing some pixel values in the image. The parameters for the Salt and Pepper noise are p and *s_vs_p*. The parameter p indicates the noise level, for instance, $p = 0.1$ indicates that the 10% of the pixel values are corrupted. On the other hand, the parameter *s_vs_p* controls the ratio between Salt and Pepper noise. For example, *s_vs_p* = 0.2 means 20% of the corrupted pixel values are white and the rest are black. The Figure 2 shows the Salt and Pepper Noise applied to an image extracted from the ORL dataset.

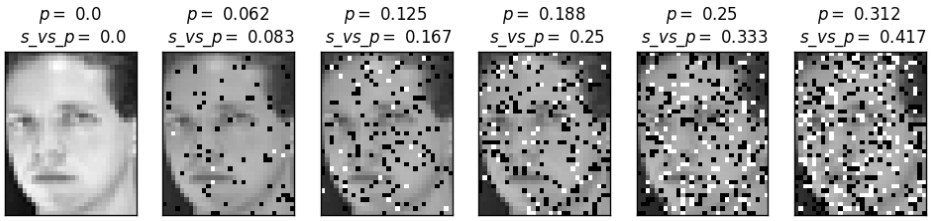

Figure 2: Image with Salt and Pepper noise with different values of hyper-parameters p and *s_vs_p*

4.2.2 Gaussian Noise

The Gaussian Noise, also known as electronic noise because usually happens in amplifiers or detectors, generates disturbs in the gray values in the digital images. It can be described as below:

$$P(g) = \sqrt{\frac{1}{2\pi\sigma^2}}^{-\frac{(g-\mu)^2}{2\sigma^2}} \tag{1}$$

where g indicates the gray value, σ the standard deviation and μ the mean. The Figure 4 shows the Gaussian Noise applied to an image extracted from the ORL dataset. Due to the properties of the normalized Gaussian noise curve, showed in the Figure 3, 70% to 90% of the noisy pixel values are between $\mu - \sigma$ and $\mu + \sigma$, considering $\mu = 0$, $\sigma = 0.1$ and 256 gray levels (g).

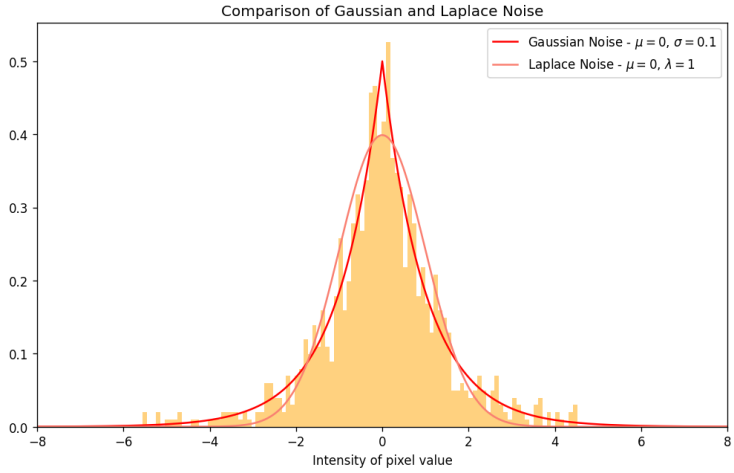

Figure 3: PDF of Gaussian and Laplace noise

4.2.3 Laplace Noise

Laplace Noise is generated by the Laplace Distribution and it is quite similar to the Gaussian Distribution although it has a sharper peak around the mean value. The Figure 3 shows the comparison between the Gaussian and the Laplace Distribution, which can be described as below:

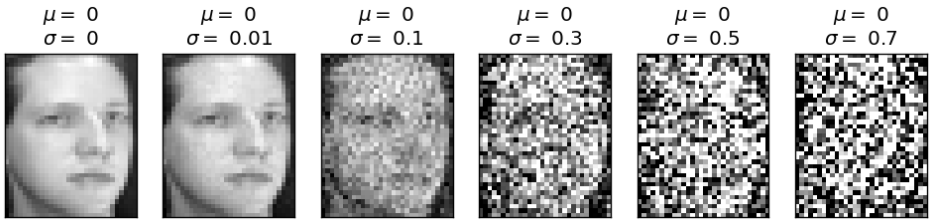

Figure 4: Image with Gaussian noise with different values of σ

$$f(x|\mu,\lambda) = \frac{1}{2\lambda}\exp(-\frac{|x-\mu|}{\lambda}) = \frac{1}{2\lambda}\begin{cases}\exp(-\frac{\mu-x}{\lambda}) & \text{if } x < \mu \\ \exp(-\frac{x-\mu}{\lambda}) & \text{if } x \geq \mu\end{cases} \tag{2}$$

where μ denotes the mean value and λ is a scale parameter. As we can observe, the probability density function is expressed in terms of the absolute difference from the mean instead of the normal distribution, which is expressed in terms of squared difference from the mean. Consequently, the Laplace distribution has thicker tails than the normal distribution. In the Figure 5, can be seen the influence of the parameter λ in an image extracted from the ORL dataset.

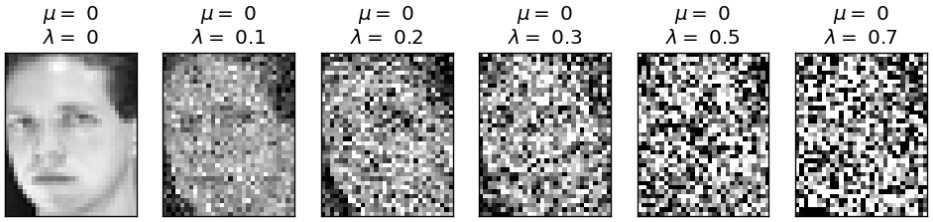

Figure 5: Image with Laplace noise with different values of λ

4.3 Metrics

To evaluate the performance of the NMF algorithms, we implement and use three evaluation metrics:

- **Relative Reconstruction Errors (RRE):** Describes the similarity between the reconstructed matrix and the original matrix. Let X denote the reconstructed matrix, and $\hat{X}$ denote the original matrix. Let U and V denote the factorization results of X, the relative reconstruction errors can be defined as follows:

$$RRE = \frac{\|\widehat{X} - UV\|_F}{\|\widehat{X}\|_F} \tag{3}$$

- **Average Accuracy:** As the images in the dataset contain n different subjects, we perform K-means clustering on the reconstructed matrix with the n clusters to get the prediction. Then, the average accuracy can be described as below:

$$ACC(Y, Y_{pred}) = \frac{1}{n}\sum_{i=1}^{n} 1\{Y_{pred}(i) == Y(i)\} \tag{4}$$

- **Normalized Mutual Information (NMI):** Can be defined by:

$$NMI(Y, Y_{pred}) = \frac{2 * I(Y, Y_{pred})}{H(Y) + H(Y_{pred})} \tag{5}$$

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

4.4 Datasets

The datasets we used to experiment are the Yale face dataset and the ORL face dataset, from each we sampled a subset to aid the speed of the experiments. The subset from Yale face dataset contains 165 images of 15 subjects under different poses and illumination conditions. We then split this into 150 images for training and 15 images for testing.

On the other hand, The ORL subdataset contains 400 images of 40 distinct people. The images were taken varying the lighting, facial expressions and facial details (some subjects wear glasses). We then split this into 360 images for training and 40 images for testing.

5 Experimental Methodology and Results

5.1 Classification with SVM

To investigate the effects of NMF as a dimensionality reduction technique, we performed face recognition using the Support Vector Machines (SVM) classifier with and without using NMF. The metrics used are the fitting and training time, the predicting time on the test set, along with the accuracy, precision, recall, F1-score of the predictions.

For each dataset, we train a SVM classifier with the train set and evaluate on the test set in 3 settings:

1. **Vanila SVM**: Train and evaluate the classifier directly as normal.
2. **L2NMF+SVM**: Use NMF with the L_2-norm to fit the training images, and use it to reduce the dimension of both the training and testing images. Then we use GridSearch to find the best hyperparameters for the SVM classifiers (search for C in range (0.00001, 0.01) and γ in range (0.0001, 0.1)). Finally we use the trained classifier to predict the labels of the test images and evaluate using the previously mentioned metrics.
3. **KLNMF+SVM**: Similar to L2NMF+SVM, but using NMF with the KL divergence.

	ORL					
Model	Fit+Train time	Predict time	Accuracy	Precision	Recall	F1-score
Vanila SVM	0+12.194s	0.020s	**0.95**	**0.93**	**0.95**	**0.93**
L2NMF+SVM	2.165+3.367s	**0.003s**	0.93	0.89	0.93	0.90
KLNMF+SVM	**0.345+3.359s**	0.004s	0.93	0.89	0.93	0.90
	Yale					
Model	Fit+Train time	Predict time	Accuracy	Precision	Recall	F1-score
Vanila SVM	0+15.959s	0.016s	**0.73**	**0.66**	**0.73**	**0.68**
L2NMF+SVM	**0.803+0.695s**	**0.001s**	0.67	0.56	0.67	0.59
KLNMF+SVM	1.176+0.794s	**0.001s**	0.67	0.56	0.67	0.59

Table 1: Table results for experiment with SVM

As can be seen in the Table 1, using NMF to reduce the dimension of the input images can significantly reduce the training and predicting time, for both of the NMF algorithms. On the other hand, this is a performance tradeoff, since the scores across all metrics slight decreases compared to the Vanila SVM classifier.

In this experiment we use SVM and supervised learning to aid the classification problem. But in the later experiments, we instead use unsupervised learning with a more primitive classifier, K-means clustering for n clusters (corresponds to n subjects), to further focus on the NMF algorithms.

5.2 Influence of Number of Components

We analyze the influence of the number of components or the reduced rank k on the performance of NMF algorithms. We based this experiment on the ORL dataset. We added noise to the images to generate a set of polluted images X and we trained the two NMF algorithms mentioned previously on this data of images. The dataset was contaminated with Gaussian Noise with a strength factor of σ = 0.05. A range of number of components values was defined from 10 to 80 with a step of 15 to train the NMF algorithms.

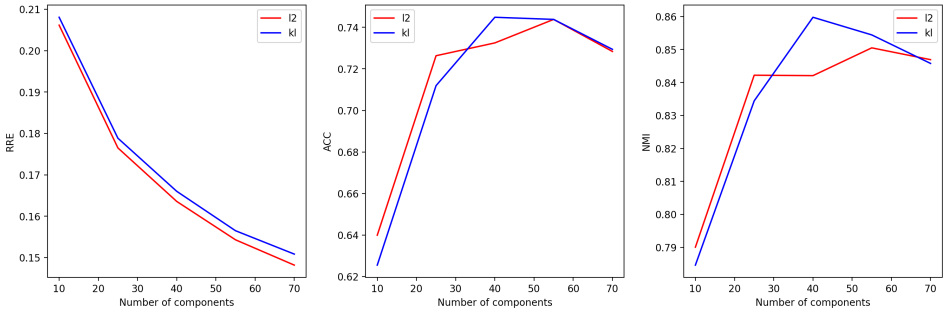

Figure 6: Influence of number of components on RRE, ACC and NMI

As we can observe in the Figure 6, the left-hand panel shows that the Relative Reconstruction Errors (RRE) decreases as the number of components k increases, which is an expected result. The more we increase the number of components, the more information is retained. Therefore, the reconstructed image given by the factorization results U and V is more similar to the original image x.

On the other hand, the middle and right figures investigate the influence between the rank k and the Average Accuracy (ACC) and Normalized Mutual Information (NMI). It can be seen that the two algorithms have a comparable performance. They achieve a maximum when k is between (40, 60) and then the ACC and NMI slightly decreases.

To examine the stability the algorithms, each experiment was executed three times and the mean and standard deviation were calculated for each experiment. Figure 7 contains the mean value and standard deviation for each metric and NMF algorithm in each experiment. It can be observed that the behavior of the two algorithms is approximately similar in each experiment. In addition, the standard deviation is generally small thus, the algorithms can be seen as stable.

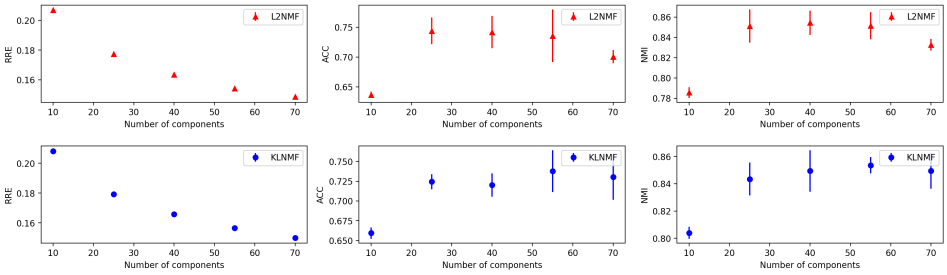

Figure 7: Mean and standard deviation of multiple runs

In general, as the number of components k increases, the performance of the NMF algorithms in terms of RRE score increase. However, in terms of ACC score and NMI score, the performance seems to reach a maximum and then starts to decrease.

5.3 Influence of Noise

We applied our three additive-noise schemes to simulate a range of real-world scenarios; Salt&Pepper, Laplacian, and Gaussian. Based on our feature selection analysis, each scenario used 60 components. Our methodology was adjusted between the two datasets (see Table 2) to account for the computational complexity. The average and standard deviation for each experiment was calculated for each metric.

	ORL			Yale		
Model	Max Iter.	No. Exp.	Repeated	Max Iter.	No. Exp.	Repeated
L2NMF	300	3	3	100	3	2
KLNMF	300	3	3	100	3	2

Table 2: Table of parameters used in the experiments

Since most of the communication and computer systems are commonly affected by specifically the Gaussian noise. Therefore, we interested in measuring the effect of the Gaussian noise scale on the face datasets. We performed experiments on different sigma values on the added Gaussian noise in order to obtain better insight on how the parameter affects our evaluation metrics.

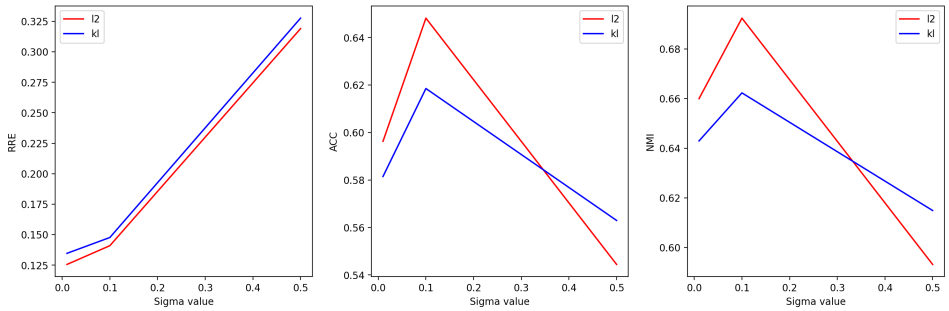

Figure 8: Influence of σ value on REE, ACC and NMI

Figure 8 indicates the mean results of the experiments on the evaluation metrics. In Gaussian noise, the σ value may be interpreted as related to the noise power. The higher the σ value, the stronger the noise in the images. It can be observed from Figure 8 that with higher σ values, the two algorithm performs worse since the presence of noise is stronger. It can also be observed the L_2 achieved higher results in all three evaluation metrics since the L_2-norm is well-suited to work with Gaussian noise.

To experiment the performance of the two NMF algorithms, we evaluate them in different configurations on the two datasets as illustrated in Table 3. In addition to the Gaussian noise investigated previously, we tested with the Salt & Pepper noise, which consist of a spread of both the percentage p (0.125 to 0.25) and range of additive noise s_v_p (0.167 to 0.333), and the Laplacian noise, with the scale factor slides linearly from 0.06 to 0.12.

As can be seen in Table 3, for the Salt&Pepper noise, the L2NMF model performs better in the Yale dataset, but has similar performance with KLNMF in the ORL dataset. In terms of the Laplacian noise, the two methods have comparable performances.

All the codes and data are available at this[1] Github repository.

6 Conclusions

In this study, we have investigated and proposed theoretical ideas to ensure the convergence property of the KL-NMF algorithm. For experiment results, the first experiment with SVM shows that NMF is a Speed/Interpretability-Performance tradeoff. The second experiment examines the effect of varying the number of components k shows that the performance reach a maximum for some k then decline, which indicates the usefulness of dimensional reduction. And the third experiment analyzes and shows the interactions between different types of noises and the 2 NMF algorithms.

Bibliography

[1] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 1999.

[2] Daniel Lee and Hyunjune Seung. Algorithms for non-negative matrix factorization. *Adv. Neural Inform. Process. Syst.*, 13, 02 2001.

[3] C. Boutsidisa and E. Gallopoulosb. Svd based initialization : A head start for nonnegativematrix factorization. 2007.

[4] Ajay Kumar Boyat and Brijendra Kumar Joshi. A review paper: Noise models in digital image processing. *ArXiv*, abs/1505.03489, 2015.

[1] `https://github.com/VanHoann/NMF_in_Facial_Recognition`

Experiment	ORL			Yale		
noise_type=S&P (p=0.125, s_v_p=0.167)	RRE	ACC	NMI	RRE	ACC	NMI
L2NMF	**0.235**	0.575	0.722	**0.237**	0.637	**0.690**
KLNMF	0.243	**0.577**	**0.728**	0.259	**0.655**	0.685
noise_type=S&P (p=0.188, s_v_p=0.25)	RRE	ACC	NMI	RRE	ACC	NMI
L2NMF	**0.272**	0.426	0.606	**0.274**	**0.637**	**0.687**
KLNMF	0.281	**0.455**	**0.620**	0.298	0.614	0.654
noise_type=S&P (p=0.25, s_v_p=0.333)	RRE	ACC	NMI	RRE	ACC	NMI
L2NMF	**0.303**	0.348	0.524	**0.303**	**0.585**	**0.634**
KLNMF	0.311	**0.355**	**0.535**	0.328	0.577	0.601
noise_type=gaussian (mean=0, sigma=0.01)	RRE	ACC	NMI	RRE	ACC	NMI
L2NMF	**0.149**	0.726	0.845	**0.125**	**0.596**	**0.659**
KLNMF	0.150	**0.729**	**0.852**	0.134	0.581	0.642
noise_type=gaussian (mean=0, sigma=0.1)	RRE	ACC	NMI	RRE	ACC	NMI
L2NMF	**0.172**	**0.719**	**0.838**	**0.140**	**0.648**	**0.692**
KLNMF	0.174	0.704	0.829	0.147	0.618	0.662
noise_type=gaussian (mean=0, sigma=0.5)	RRE	ACC	NMI	RRE	ACC	NMI
L2NMF	**0.377**	**0.234**	0.423	**0.319**	0.544	0.593
KLNMF	0.387	0.232	**0.430**	0.327	**0.562**	**0.614**
noise_type=laplace (loc=0, scale=0.06)	RRE	ACC	NMI	RRE	ACC	NMI
L2NMF	**0.158**	0.680	0.820	**0.134**	0.622	0.682
KLNMF	0.162	**0.706**	**0.841**	0.139	**0.670**	**0.704**
noise_type=laplace (loc=0, scale=0.09)	RRE	ACC	NMI	RRE	ACC	NMI
L2NMF	**0.182**	0.662	**0.810**	**0.148**	0.614	0.659
KLNMF	0.187	**0.677**	0.804	0.152	**0.644**	**0.678**
noise_type=laplace (loc=0, scale=0.12)	RRE	ACC	NMI	RRE	ACC	NMI
L2NMF	**0.208**	**0.647**	**0.777**	**0.163**	0.622	0.686
KLNMF	0.213	0.632	0.774	0.168	**0.637**	**0.688**

Table 3: Table of results for Noise experiment