# Compare Robustness of Nonnegative Matrix Factorization Algorithms

Chen Chen 480458339
Xiaodan Gan 440581983
Xinyue Wang 440359463

8th April 2019

**Abstract**

*This project numerically compares the robustness of two nonnegative matrix factorisation (*NMF*) algorithm [**?**] based on multiplicative update rules when contaminated by large magnitude Gaussian, Poisson and Salt & Pepper noise. Section 2 briefly reviews relevant work in the field of* NMF*, which found these multiplicative update rules are very sensitive to initialisation. Section 3 describes the two algorithms and their theoretical properties, proposes a solution to make the multiplicative update rules less sensitive to initialisation, and details the statistical tools we used to compare the robustness of the two algorithm. Our simulation results in Section 4 agree with the theoretical properties of the two algorithms. Future work involves embedding the proposed multiple initialisations into the* NVIDIA GPU *based parallel-programming model.*

# Contents

**1 Introduction** **1**

**2 Related work** **2**

**3 Methods** **3**
3.1 NMF and Gaussian noise . . . . . 3
3.2 KLNMF and Poisson noise . . . . . 4
3.3 Preprocess . . . . . 5
3.4 Gaussian and Poisson are asymptotic equivalent . . . . . 5
3.5 Salt & Pepper noise . . . . . 6

3.6 Multiple initial estimates assure the algorithms stable . . . . . 6
3.7 KLNMF requires more iterations . . . . . 7
3.8 Evaluation metrics and their confidence intervals . . . . . 7
3.9 Statistical method compares the robustness of algorithms . . . . . 7

**4 Experiments** **8**
4.1 Dataset . . . . . 8
4.2 Noise . . . . . 8
4.2.1 Gaussian Noise . . . . . 9
4.2.2 Poisson Noise . . . . . 9
4.2.3 Salt & Pepper Noise . . . . . 9
4.3 Experiment Setup . . . . . 10
4.4 Experiments Results . . . . . 10
4.4.1 Two algorithms output similar reconstructed images . . . 10
4.4.2 Hypothesis test distinguishes the difference in RRE . . . . 11
4.4.3 ACC and NMI results . . . . . 12
4.5 Personal Reflection . . . . . 12

**5 Conclusion** **14**

# 1 Introduction

Non-negative matrix factorisation (NMF) is a matrix decomposition technique that approximates a data matrix with non-negative entries with the multiplication of two non-negative matrices

$$V \approx WH.$$

In this approximation, matrix $W$ is the basis, and matrix $H$ is the weight matrix corresponding to the new dictionary $W$. NMF is applicable to an extensive range of domain. For example, **?** suggest that NMF is useful for image processing problems including facial recognition.

Matrices $W$ and $H$ are often referred as the basis images and weights, because the observed image $V$ is approximated by a linear combination of $W$ with positive coefficients $H$. Due to the additive nature of the algorithm, the dictionary $W$ are often parts of images that are easily interpretable. This property also distinguishes NMF from other traditional image processing methods such as principal components analysis (PCA). **?** demonstrate that NMF performs better in image classification problems in comparison with PCA. Moreover, NMF is also applicable to text mining such as semantic analysis. Generally, NMF is useful to discover semantic features

of an article by counting the frequency of each word, and then approximating the document from a subset of a large array of features [**?**].

In practice, face images could be easily corrupted during data collection by noise with large magnitude. Corruption may result from lighting environment, facial expression or facial details. An NMF algorithm that is robust to large noise is desired for the real-world applications. Therefore, the objective of this project is to analyse the robustness of NMF algorithms on corrupted datasets. We implement two NMF algorithms proposed by **?** on real face image datasets, `ORL` dataset and the `CroppedYale` dataset. We add artificial noises to the face images are contaminated. We then apply nonparametric statistical methods to analyse the robustness of these two algorithms from simulation results.

# 2 Related work

Researchers proposed many NMF algorithms. As the objective function of NMF is non-convex, for which the traditional gradient decent method could be very sensitive to step sizes and converge slowly, **?** first propose to algorithms which minimise Euclidean distance or Kullback-Leibler divergence (KLNMF) between the original matrix and its approximation. Although these algorithms are easy to implement and have reasonable convergent rate [**?**], they require more iterations than alternatives such as gradient descent [**?**]. Also, the algorithms may fail on seriously corrupted datasets which violate its assumption of Gaussian noise or Poisson noise, respectively [**?**]. Moreover, **?** indicate that these methods are sensitive to the initial selection of matrices $W$ and $H$. The algorithms require many iterations to retrieve from poorly selected initial values.

Apart from different loss functions, several optimisation methods were proposed to improve the performance of NMF. After **?** proposed multiplicative update rules MUR, **?** proposed a modified MUR which guaranteed the convergence to a stationary point. This modified MUR, however, did not improve the convergence rate of traditional MUR [**?**]. Moreover, as MUR does not impose sparseness, **?** proposed a projected nonnegative least square (PNLS) method to enforce sparseness. In each nonnegative least square sub-problem, this algorithm projects the negative elements of least squares solution directly to zero. Nevertheless, PNLS does not guarantee convergence [**?**].

In contrast to these gradient-based optimisation methods, **?** presented an active set method which divides variables into two sets, a free set and an active set. They update free set in each iteration by solving an unconstrained equation. Even though

the active set method has a nice convergence rate, it assumes strictly convexity in each nonnegative least square sub-problem [**?**]. These assumptions are easily violated in real life applications.

There exist many robust NMF algorithms which include penalties in the objective functions. For example, **?** proposes $L_1$-norm based NMF to model noisy data with a Laplace distribution which is less sensitive to outliers. However, as $L_1$-norm is not differentiable at zero, the optimisation procedure is computationally expensive. **?** proposed an NMF algorithm using $L_{21}$-norm loss function which is more stable. The updating rules used in $L_{21}$-norm NMF, however, still converge slowly because of continual use of the power method [**?**].

# 3 Methods

Some carefully designed NMF are robust to various noises. These robust algorithms aim to significantly reduce the amount of noise while preserving the edges without blurring the images [**?**].

## 3.1 NMF and Gaussian noise

**Gaussian noise** is noise with a probability density function being normal with mean zero. **?** propose the first NMF with the objective function between image $V$ and its NMF factorisation $W$ and $H$ being

$$\|V - WH\| = \sum_{ij} \left[V_{ij} - (WH)_{ij}\right]^2. \tag{1}$$

To minimise this object function of least square, **?** prove the convergence of the multiplication update rule

$$H_{jk} \leftarrow H_{jk} \frac{(W^T V)_{jk}}{(W^T WH)_{jk}} \text{ and } W_{ij} \leftarrow W_{ij} \frac{(VH)_{ij}}{(WHH^T)_{ij}}. \tag{2}$$

Here, $()_{ij}/()_{ij}$ denotes an entry-wise division of the two matrices. **?** show this NMF algorithm minimises Gaussian.

## 3.2 KLNMF and Poisson noise

**Poisson noise** or shot noise is a type of electronic noise that occurs when the finite number of particles that carry energy, such as electrons in an electronic

circuit or photons in an optical device, is small enough to give rise to detectable statistical fluctuations in a measurement. **?** suggest that KLNMF is an algorithm that minimising the Kullback-Leibler divergence

$$\begin{aligned} D(V||WH) &= \sum_{ij}\left(V_{ij}\log\frac{V_{ij}}{(WH)_{ij}} - V_{ij} + (WH)_{ij}\right) \\ &= \sum_{ij}\left(-V_{ij}\log(WH)_{ij} + (WH)_{ij} + C(V_{ij})\right). \end{aligned} \tag{3}$$

where $C(V_{ij}) = V_{ij}\log V_{ij} - V_{ij}$. $C(V_{ij})$ is a function of the observed image matrix $V$ only. **?** also suggest a multiplication update rule to find as the optimisation procedure of KLNMF

$$H_{jk} \leftarrow H_{jk}\frac{\sum_i W_{ij}V_{ik}/(WH)_{jk}}{\sum_{i'} W_{i'j}} \text{ and } W_{ij} \leftarrow W_{ij}\frac{\sum_k H_{jk}V_{ik}/(WH)_{jk}}{\sum_{k'} H_{ik'}}. \tag{4}$$

As this original image matrix $V$ is observed, minimising this Kullback-Leibler divergence (3) is equivalent to minimising

$$\sum_{ij}\left(-V_{ij}\log(WH)_{ij} + (WH)_{ij} + C(V_{ij})\right).$$

, for arbitrary bounded function $C(V_{ij})$. Taking exponential of the negative of this score function, the problem transforms to maximising the following likelihood function

$$L(WH|V) = \prod_{ij}\left((WH)_{ij}^{V_{ij}}\, e^{-(WH)_{ij}} + C(V_{ij})\right).$$

Choosing constant $C(V_{ij})$ to be $-\log V_{ij}!$ gives

$$L(WH|V) = \prod_{ij}\left(\frac{(WH)_{ij}^{V_{ij}}\, e^{-(WH)_{ij}}}{V_{ij}!}\right).$$

Hence, the probability density function of each element of the original matrix V is Poisson

$$P(V_{ij}) = \frac{(WH)_{ij}^{V_{ij}}\, e^{-(WH)_{ij}}}{V_{ij}!}$$

is a sufficient condition to yield this likelihood. Hence KLNMF is most suitable for images with Poisson noise.

## 3.3 Preprocess

We did not preprocess the images because both of NMF and KLNMF are scale sensitive, because both objective functions (1) and (3) varies when original matrix $V$ and result matrices $W$ and $H$ scale proportionally, i.e. for $\lambda$ real, $D(V||WH) \neq D(\lambda V||\lambda WH)$. To overcome this issue, we could have normalised the matrices $W$ and $H$ in each iteration [**?**], but it will result in an even slower computation.

## 3.4 Gaussian and Poisson are asymptotic equivalent

We design a Gaussian noise and a Poisson noise with different magnitude. Poisson distribution with parameter $\lambda$ (integer) is equivalent to the sum of $\lambda$ Poisson distributions with parameter 1 [**?**, p. 45]. Hence for $\lambda$ large, Central Limit Theorem implies that Poisson distribution with parameter $\lambda$ is well approximated by $N(\lambda, \lambda)$. When applying Poisson noise to an image, we do not have degree of freedom to choose any parameter. The variance is the magnitude of the pixels. To compare the robustness of KLNMF with NMF with different noise, we choose the variance of Gaussian noise to be the difference from the magnitude of the pixel, that is, $N(0, \text{Var}) \neq N(0, V) \approx \text{Poi}(V) - V$. Figure 1 visualises the similarity of Poisson distribution and Normal distribution with parameter $V = 40$. To overcome this issue, the Gaussian noise we use should have very different variance in comparison with the mean of the images. The pixel mean of the ORL dataset is approximately 40, and the pixel mean of the Cropped Yale set is approximately 70. Hence, we choose the variance of the noise to be $80^2$ so that it is the way bigger than 255, which is the maximum value of a pixel.

## 3.5 Salt & Pepper noise

Apart from Gaussian and Poisson noises, we also test our two algorithms on the commonly seen **Salt & Pepper noise**. The noise presents itself by having dark pixels in bright regions and bright pixels in dark regions [**?**].

## 3.6 Multiple initial estimates assure the algorithms stable

As discussed in the section of related work, The problem of nonnegative matrix factorization is not a convex problem. Hence the results update rules (2) and (4) coverage to maybe local minima instead of global minima, depending on the initial approximation. Our task was to compare the robustness of the algorithm, and

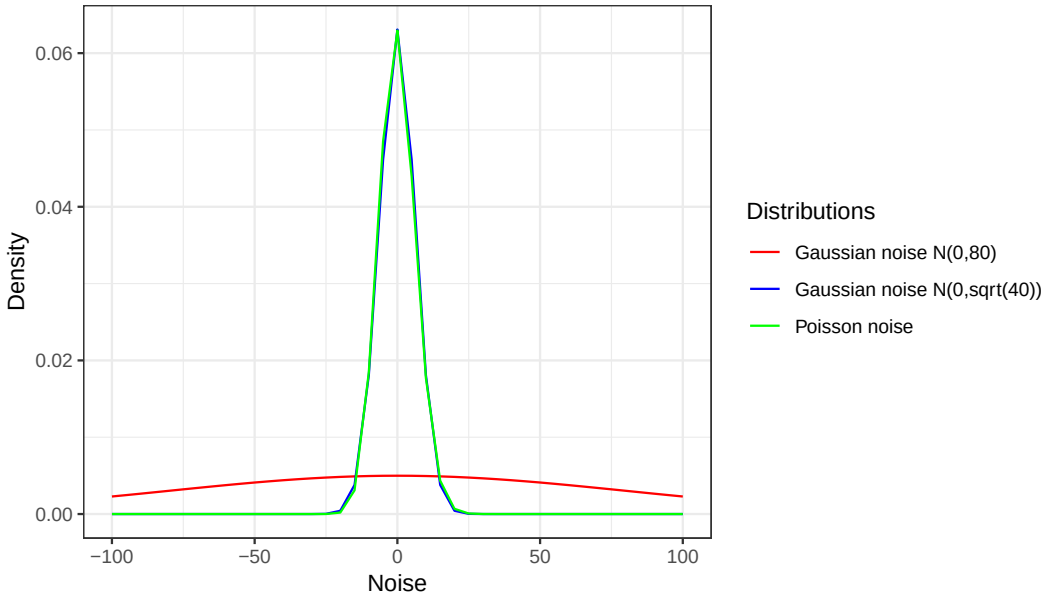


Figure 1: Compare a Gaussian noise $\mathrm{N}(0,40)$ with Poisson noise $\mathrm{Poi}(40)-40$. They two distributions are asymptotically equivalent and have overlapped density functions. The Gaussian noise $\mathrm{N}(0,80^2)$ is very different.

we do not want the instability of our algorithms to affect our comparison. To address this issue, we implement several (i.e. $n$) initial estimates for each matrix factorisation problem. We use the factorised matrices $W$ and $H$ corresponding to the least residual (1) and (3), for NMF and KLNMF, algorithms respectively, as the final result of factorisation. This design of multiple starting point improves the stability of the algorithms, but it requires more computational power. To improve the computational speed, we make the number $n$ equal to the number of cores of the CPU. We assign each of the $n$ initial estimates randomly with a uniform distribution. Then these each of the $n$ initial estimates is assigned to a different core of the CPU. This boosts the CPU utilisation to 100% instantly and improved the computational speed by 70% on the `ORL` data. The following part of our code implements this idea of parallel computing.

Algorithm 1: Multi-start paralle computing

```
args = zip(repeat(V,ncpu), repeat(r,ncpu),
    repeat(niter[name2],ncpu), repeat(min_error[name2],ncpu))
result = pool.starmap(algo, args)
```

where `algo` is the NMF algorithm and `niter` is the number of iterations. We use a 16-thread Xeon high performance computer to run this algorithm. The algorithms

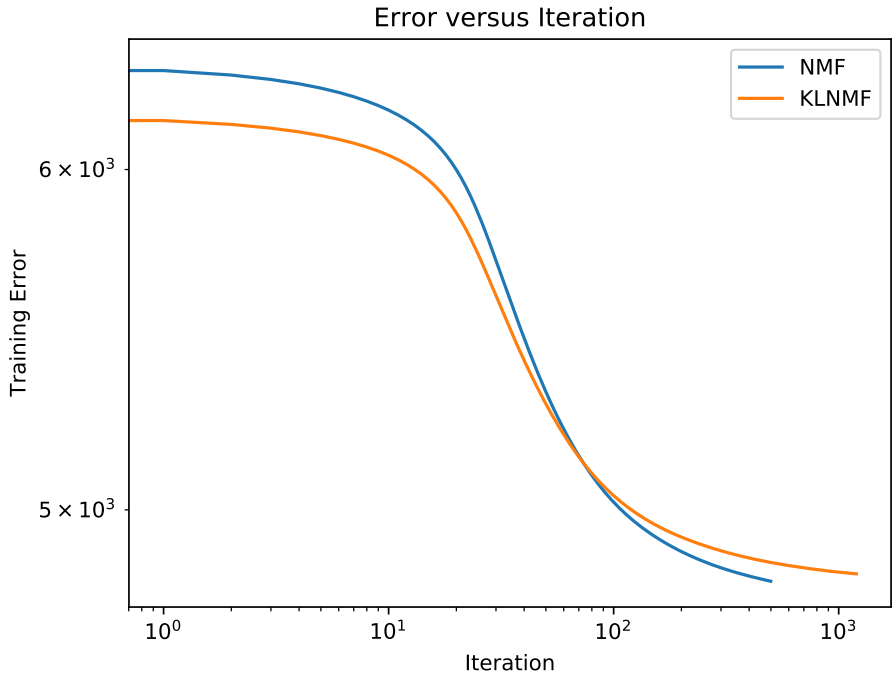


Figure 2: Residual of the objective function (1) and (3) versus the number of iterations. NMF converges more than twice faster comparing with KLNMF.

run in this high performance computer so that the computing time is reasonable. The multiple initial estimates assure the algorithms are stable.

## 3.7 KLNMF requires more iterations

A residual versus the number of iteration plot (Figure 2) shows that KLNMF converges slower than NMF. In the log-log plot, the slope of the NMF residual plot is 2.5 times larger than that of the KLNMF plot for the ORL data. The slope estimates that the rate of convergence of NMF is 2.5 faster than KLNMF. As a result, we set the number of iterations as 500 and 1200 for NMF and KLNMF algorithms, respectively (i.e. roughly 2.5 more iterations).

## 3.8 Evaluation metrics and their confidence intervals

The assignment instruction asks us to compare the performance of NMF and KLNMF by using evaluations metrics including Relative Reconstruction Errors (RRE), Average Accuracy (AA), Normalized Mutual Information (NMI). The instruction states the formulae of these metrics. However, to systematically compare the metrics, we construct an 95% confidence interval for any metric (e.g. RRE) by bootstrapping percentile confidence interval. The idea of bootstrapping is straightforward—we resample a subset of 40 samples among the sample space of 80 Monte-Carlo simulations and calculate the mean. We repeat this process 1000 times. The 2.5% and 97.5% percentiles of the 80 resampled means are then the bootstrapping percentile confidence interval

$$(\mathrm{RRE}^*_{2.5}, \mathrm{RRE}^*_{97.5}), \tag{5}$$

where $\mathrm{RRE}^*_\alpha$ is the $\alpha$ percentile of the bootstrapped distribution from our sample space with 80 Monte-Carlo simulations. We run 80 simulations so that the confidence interval we construct is precise.

Bootstrapping does not require the sample space follows specific distributions. We apply this nonparametric method to construct confidence interval here because we do not know about the exact distributions of these three evaluation metrics.

## 3.9 Statistical method compares the robustness of algorithms

We implement the Kolmogorov-Smirnov test to test the hypothesis that the algorithms NMF and KLNMF have different robustness. Again Kolmogorov-Smirnov test is distribution free, so we do not need to know the distributions of the evaluation metrics.

Let $\mathrm{RRE}_i$ denote the RRE generated from our NMF algorithm by the $i$th Monte-Carlo simulation. Define the empirical distribution of a sample set generated by algorithm $\alpha$, perhaps the 80 RRE results, as

$$\hat{F}_\alpha(x) = \frac{1}{n}\sum_{i=1}^{80} 1_{\mathrm{RRE}_i \le x}. \tag{6}$$

The test statistic is the supremum among the differences of the empirical distribution generated using definition (6) [**?**]

$$D = \sup_x |F_{\alpha_2}(x) - F_{\alpha_2}(x)|. \tag{7}$$

We compare the test statistics $D$ with the critical value of $0.215$, which corresponds to 80 samples and a 95% of confidence level. We reject the null hypotheses that the two algorithms produce similar RRE (or other evaluation metrics) with 95% confidence level if the test statistics $D > 0.215$. The technique is extended to compare AA and NMI.

# 4 Experiments

## 4.1 Dataset

We illustrate our two NMF algorithms on two real-world face image datasets: `ORL` and `CroppedYaleB` (**?**). Both `ORL` and `CroppedYale` datasets contain multiple images of distinct subjects with various facial expression, lighting condition, and facial details. Images in ORL are cropped and resized to $92 \times 112$ pixels. We further rescale it to $30 \times 37$ pixels. Similarly, we reduce the size of images in `CroppedYale` to $42 \times 48$ pixels. For each dataset, we flatten the image matrix into a vector and append them together to get a matrix $V$ with shape $d \times n$ where integer $d$ is the number of pixels in one image and integer $n$ is the number of images. In each epoch, we use 90% of data.

## 4.2 Noise

We implement three kinds of noises including Gaussian noise, Poisson noise and Salt & Pepper noise.

### 4.2.1 Gaussian Noise

We design the Gaussian noise by a normal distribution with a mean of $0$ and a standard deviation of $80$ (Algorithm 2). The `ORL` dataset has a global pixel mean of 40 and the `CroppedYale` dataset has that of 70. Hence the designed Gaussian noise contaminates the images significantly. We choose the standard deviation to be 80 so that our Gaussian noise is less likely to coincident with the designed Poisson noise. To satisfy the nonnegative constant, negative value in the contaminated image is set to zero.

Algorithm 2: Gaussian Noise Design

```
def normal(subVhat):
    """Design a Gaussian noise."""
```

```
    V_noise = np.random.normal(0, 80, subVhat.shape) #*
        np.sqrt(subVhat)
    V = subVhat + V_noise
    V[V < 0] = 0
    return V, V_noise
```

### 4.2.2 Poisson Noise

The Poisson noise is not additive and has no hyperparameters to be set. Unlike Gaussian noise, contaminated images are drawn directly from Poisson distributions with parameters set to be pixel values. Then, the Poisson noise is calculated from the difference between the contaminated image and the original image, as discussed in Section 3 and demonstrated in Algorithm 3.

Algorithm 3: Poisson Noise Design

```
def possion(subVhat):
    """Design a Possion noise."""
    V = np.random.poisson(subVhat)
    V_noise = V-subVhat
    return V, V_noise
```

### 4.2.3 Salt & Pepper Noise

Salt & Pepper noise (Algorithm 4) is added by drawing random integers from the discrete uniform distribution of the interval $[0, 255)$ . We find the bright places in generated image and replace pixel values in the same place of the original image with the brightest value. Similarly, we also find the dark pixels in the images and replace pixel values in the same place of original image with the darkest pixel value. In this case, we set the pixels whose values are greater than or equal to 230 as bright pixels and pixels whose value being less than or equal to 20 as dark pixels.

Algorithm 4: Salt and Pepper Noise Design

```
def salt_and_pepper(subVhat):
"""Design a salt and pepper noise where make some pixel value
    zeros."""
  V_noise = np.random.randint(low=0, high=255,
      size=subVhat.shape, dtype=int)
  V = subVhat.copy()
  V[V_noise <= 20] = 0
  V[V_noise >= 230] = 255
  return V, V_noise
```

## 4.3 Experiment Setup

We apply two algorithms (NMF and KLNMF) with four categories of noises (no noise, Gaussian noise, Poisson noise and Salt & Pepper noise), which results in eight combinations in each epoch. In each epoch, we randomly select 90% of samples to train NMF algorithms and evaluate three metrics on reconstructed images. The training will terminate when the error reaches the minimum error, or the maximum iteration is reached. The minimum error and maximum iteration are hyperparameters which we learn from iterative experiments. Our code saves the learning errors versus the number of iterations so that we could draw the plot and observe the convergence of the learning process. We increase the number of epochs and calculate the average metrics and confidence intervals.

## 4.4 Experiments Results

### 4.4.1 Two algorithms output similar reconstructed images

Figure 3 and 4 visualise the original image, designed noises, corrupted images and reconstructed images from left to right. From top to bottom, the four rows correspond to no noise, Gaussian noise discussed in Section 4.2.1, Poisson noise discussed in Section 4.2.2 and Salt & Paper noise discussed in Section 4.2.3. The first row of Figure 3 and 4 show both algorithms reconstructed the original image well without artificial noise and with Poisson noise. However, when the noise is large (the second and last rows in Figure 3 and 4), the quality of reconstructed images looks marginally better than the contaminated images. This result is consistent with **?**, who assert that NMF may fail to handle extremely corrupted images when we violate the assumptions on the distributions of noise. Moreover, the difference between images generated by NMF and KLNMF are not visually significant. Hence, we implement the statistical hypothesis test to compare of RREs of the two algorithms.

### 4.4.2 Hypothesis test distinguishes the difference in RRE

Substitute RRE results (Figure 5) from 80 Monte-Carlo simulations of the two algorithms into Kolmogorov-Smirnovs test (7) gives test statistics $D = 1, 1, 0.6625$ with no noise, Gaussian noise, and Poisson noise, for the ORL dataset. These three test statistics are all much greater than the critical value 0.215. Hence there are strong statistical evidences that the performance of NMF and KLNMF are different in these three problems. For salt and Pepper noise, test statistic $D = 0.2125 < 0.215$,

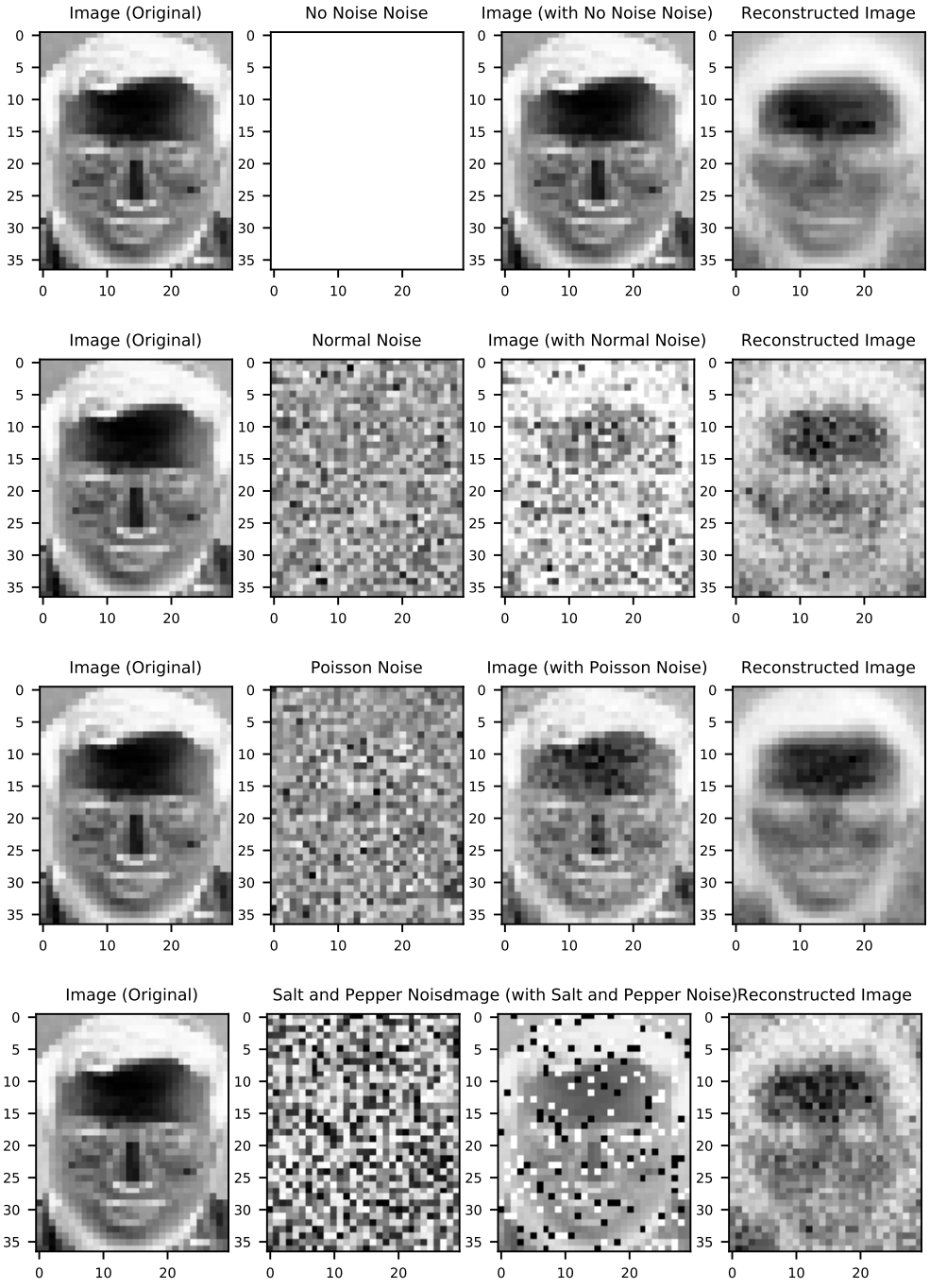


Figure 3: The reconstructed image by NMF. The original images (Column 1) are combined with noises (Column 1) including Gaussian Noise with Variance 80 (Row 2), Poisson Noise (Row 3), and Salt & Pepper Noise (Row 4). The corrupted images are shown in Column 3. The reconstructed images are shown in (Column 4). The reconstruction with no noise is shown in Row 1.

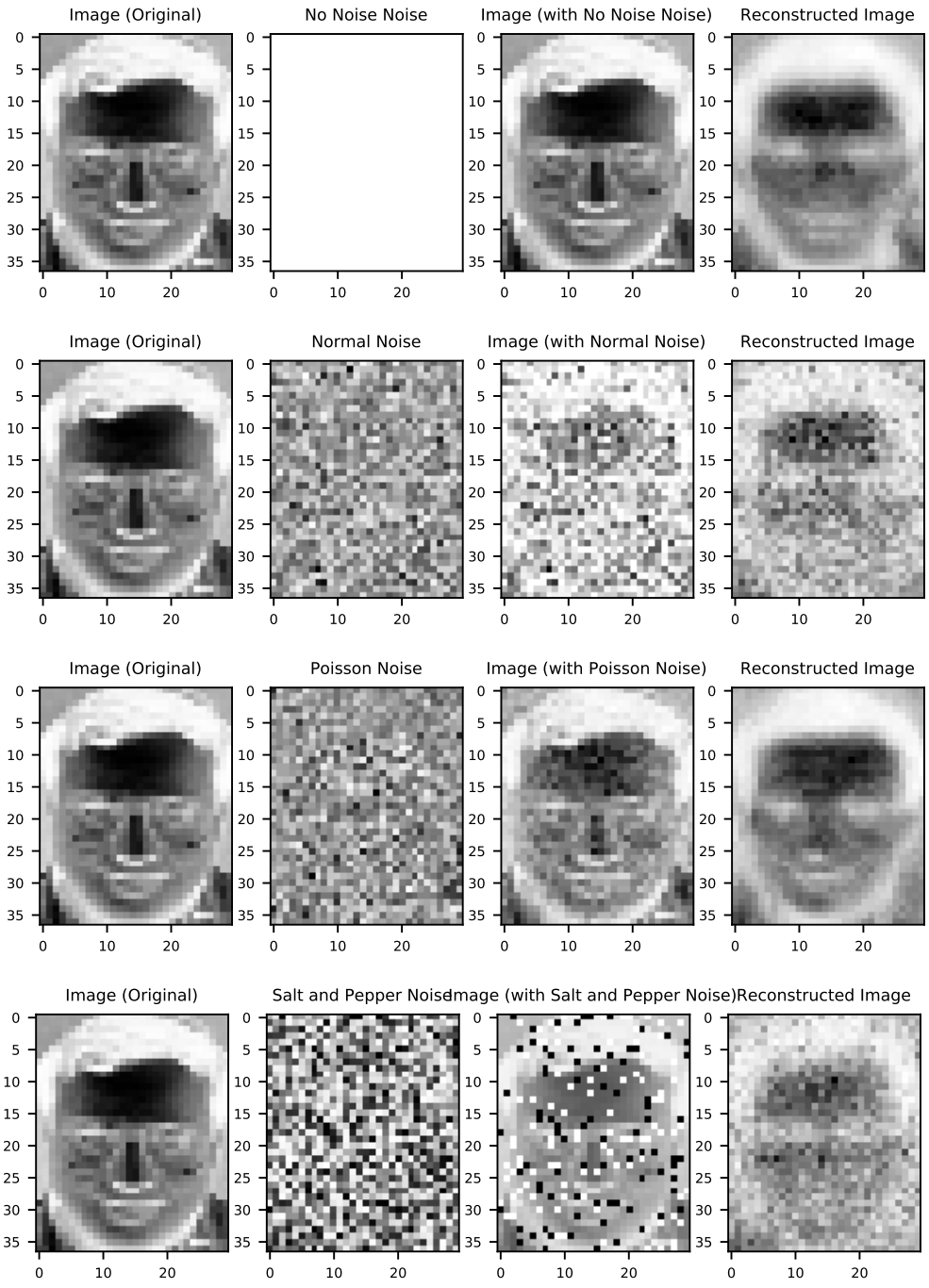


Figure 4: The reconstructed image by KLNMF. The original images (Column 1) are combined with noises (Column 1) including Gaussian Noise with Variance 80 (Row 2), Poisson Noise (Row 3), and Salt & Pepper Noise (Row 4). The corrupted images are shown in Column 3. The reconstructed images are shown in (Column 4). The reconstruction with no noise is shown in Row 1.

C Chen, X Gan, X Wang, 8th April 2019

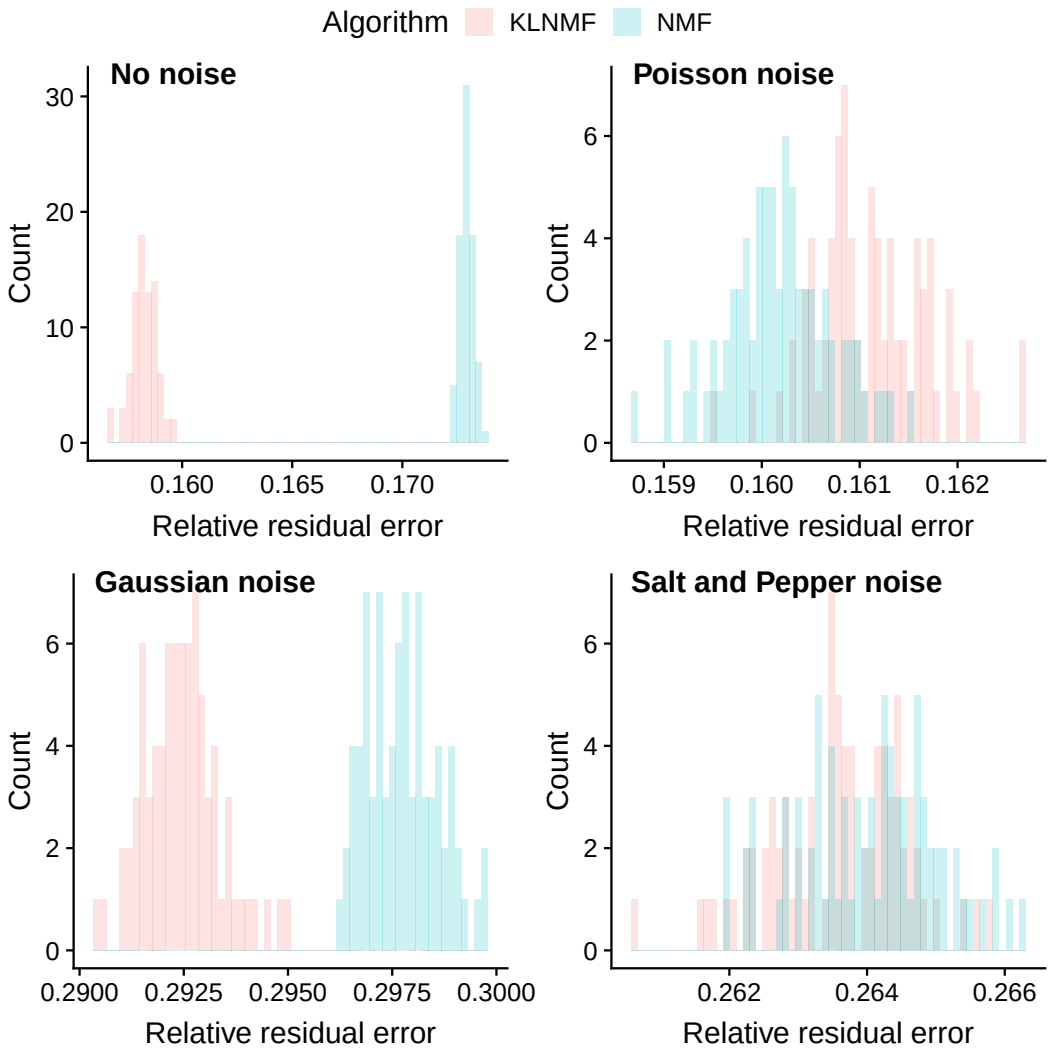


Figure 5: Histogram of RRE results from 80 Monte-Carlo simulations. Blue bars correspond to NMF, and the pink bars correspond to KLNMF. The types of noise are labelled in the plot. The visualisation agrees with our statistical analysis.

hence we fail to conclude that the two methods have different robustness against Salt and Pepper noise. Further, one tail Kolmogorov-Smirnov test concludes that KLNMF performs better reconstructing the original image with no noise and is more robust Poisson noise. In contrast, NMF is more robust against Gaussian noise, even with only 500 iterations (1200 for KLNMF).

Theoretical results discussed in Section 3 suggest NMF is more robust against Gaus-

Table 1: Average of evaluations metrics over 80 Monte-Carlo simulations using the ORL dataset. The 95% confidence intervals are calculated using bootstrap.

| ORL dataset | RRE | ACC | NMI |
|---|---|---|---|
| NMF no noise | 0.1583 (0.1581, 0.1584) | 0.7364 (0.731, 0.742) | 0.8536 (0.8506, 0.8567) |
| NMF Gaussian noise | 0.2925 (0.2922, 0.2927) | 0.447 (0.4423, 0.4521) | 0.6212 (0.6176, 0.6247) |
| NMF Poisson noise | 0.1611 (0.161, 0.1613) | 0.7313 (0.7262, 0.7367) | 0.8493 (0.8456, 0.8527) |
| NMF Salt and Pepper noise | 0.2636 (0.2634, 0.2638) | 0.5094 (0.504, 0.5151) | 0.6721 (0.6679, 0.6764) |
| KLNMF no noise | 0.1729 (0.1728, 0.173) | 0.7406 (0.7352, 0.7458) | 0.8599 (0.8568, 0.8632) |
| KLNMF Gaussian noise | 0.2977 (0.2976, 0.2979) | 0.4538 (0.4483, 0.4595) | 0.6209 (0.6165, 0.6255) |
| KLNMF Poisson noise | 0.1602 (0.1601, 0.1603) | 0.7417 (0.7365, 0.7472) | 0.8573 (0.8542, 0.8602) |
| KLNMF Salt and Pepper noise | 0.264 (0.2638, 0.2643) | 0.5089 (0.5038, 0.5139) | 0.6734 (0.6694, 0.6779) |

sian noise whereas KLNMF is more robust against Poisson noise. Our experimental results concluded from Kolmogorov-Smirnovs hypothesis tests agree with these theoretical results. Further, as both of the algorithms are not designed for Salt and Pepper noise, they have a similar performance against it.

These results can be observed by directly reading whether the confidence intervals overlap in Table 1. Also, the statistical results agree with the visualisation in Figure 5, 3 and 4. These results also agree with our intuition—although the differences in the robustness of the two algorithms are small, the even smaller variances in the RRE results make them statistically different under Poisson and Gaussian noise (Figure 5).

With the CroppedYale dataset, the test results (Table 2) agree with those of the ORL data as well as theories in Section 3: 1. against Poisson noise, KLNMF with update rule (4) has a lower RRE; 2. against Gaussian noise, NMF with update rule (2) has a lower RRE; 3. KLNMF has a lower RRE for images with no artificial noise. However, using CroppedYale dataset, the NMF algorithm performs much better against Salt & Pepper noise. This might be a result of the NMF's much faster convergence rate for large dataset, that is, the performance of KLNMF might be significantly better with more iterations when processing CroppedYale. We did not perform more iterations for KLNMF due to running time constraint.

### 4.4.3 ACC and NMI results

In terms of ACC and NMI, the differences between NMF and KLNMF under Gaussian noise and Salt & Pepper noise are trivial given their overlapped confidence intervals. Under Possion noise, however, KLNMF is superior than NMF for both measurements.

Table 2: Average of evaluations metrics over 10 Monte-Carlo simulations using `CroppedYale` dataset.

| `CroppedYale` dataset | RRE | ACC | NMI |
|---|---|---|---|
| NMF no noise | 0.2022 | 0.2377 | 0.3204 |
| NMF Gaussian noise | 0.3114 | 0.2116 | 0.2777 |
| NMF Poisson noise | 0.2023 | 0.2442 | 0.3241 |
| NMF Salt and Pepper noise | 0.2748 | 0.2133 | 0.2902 |
| KLNMF no noise | 0.2062 | 0.2434 | 0.3191 |
| KLNMF Gaussian noise | 0.3106 | 0.2112 | 0.2845 |
| KLNMF Poisson noise | 0.2065 | 0.2377 | 0.3104 |
| KLNMF Salt and Pepper noise | 0.3164 | 0.1634 | 0.2332 |

## 4.5 Personal Reflection

The implementation of this project was challenging and rewarding. We overcome many problems that are not taught in class by research. For instance, we learnt to use multi-start algorithm with parallel computing to overcome the problem that KLNMF always gets stuck into local optimal. In addition, we learnt that although one algorithm performs better than another one theoratically, it might require more running time. In real implementation, we need to consider the running time versus performance trade-off. For example, the KLNMF multi-start parallel computing has overall better performance than NMF; however, it has slow convergence rate and multi-start parallel computing will cost more than than non-multi-start KLNMF algorithm, especially when training `CroppedYale` dataset . Moreover, we critically considered what truly defines a 'good' algorithm. Although theoretically privileged algorithms may be good for handling difficult tasks, they tend to be time-consuming and not easy to implement. For example, through literature review, we found some algorithms such as Truncated Cauchy NMF **?** that are excellent for contaminated data but too difficult for us to implement. Hence, in real-world practice, simpler and faster algorithm may be more widely used than advanced algorithms. Lastly, we observed many interesting results during the experiment. For instance, we found that the RREs of KLNMF with Poisson noise is superior to that with no noise in `ORL` dataset. This may result from the noise assumptions made by these two algorithms.

# 5 Conclusion

In conclusion, our numerical simulation supports the theoretical results that NMF based on objective function (1) is more robust to Gaussian noise and KLNMF based on objective function (3) is more robust to Poisson noise. The two algorithms have a similar performance against Pepper & Salt noise. However, the NMF algorithm reconstructs image better without noise and converges much faster with the multiplicative update rule when comparing with KLNMF. Also, simulation results find both of the multiplicative update rules are sensitive to the initial values of matrices $W$ and H. Section 3 proposes a solution based on parallel programming to solve this problem. However, this solution requires high performance computer so that the algorithm converges in a reasonable amount of time.

Recently, NVIDIA released their Cuda package which parallelises algorithms using GPU. This package improves the speed of parallelisable algorithms, including many NMF algorithms, by a factor of $> 100$. As a computationally expensive procedure, our suggestion of using multiple initialisations will be more novel if a GPU version based on Cuda could be designed.

## References

Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 556–562. MIT Press, 2001. URL http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf.

Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788, 1999.

David Guillamet and Jordi Vitrià. Non-negative matrix factorization for face recognition. In *Topics in artificial intelligence*, pages 336–344. Springer, 2002.

Michael W Berry, Murray Browne, Amy N Langville, V Paul Pauca, and Robert J Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational statistics & data analysis*, 52(1):155–173, 2007.

Naiyang Guan, Tongliang Liu, Yangmuzi Zhang, Dacheng Tao, and Larry Steven Davis. Truncated cauchy non-negative matrix factorization for robust subspace learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Zhirong Yang, He Zhang, Zhijian Yuan, and Erkki Oja. Kullback-leibler divergence for nonnegative matrix factorization. In *International Conference on Artificial Neural Networks*, pages 250–257. Springer, 2011.

Chih-Jen Lin. On the convergence of multiplicative update algorithms for nonnegative matrix factorization. *IEEE Transactions on Neural Networks*, 18(6): 1589–1596, 2007.

Naiyang Guan, Dacheng Tao, Zhigang Luo, and Bo Yuan. Nenmf: An optimal gradient method for nonnegative matrix factorization. *IEEE Transactions on Signal Processing*, 60(6):2882–2898, 2012.

Hyunsoo Kim and Haesun Park. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM journal on matrix analysis and applications*, 30(2):713–730, 2008.

Edmund Y Lam. Non-negative matrix factorization for images with laplacian noise. In *Circuits and Systems, 2008. APCCAS 2008. IEEE Asia Pacific Conference on*, pages 798–801. IEEE, 2008.

Deguang Kong, Chris Ding, and Heng Huang. Robust nonnegative matrix factorization using l21-norm. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 673–682. ACM, 2011.

Tudor Barbu. Variational image denoising approach with diffusion porous media flow. In *Abstract and Applied Analysis*, volume 2013. Hindawi, 2013.

Tongliang Liu and Dacheng Tao. On the performance of manhattan nonnegative matrix factorization. *IEEE Transactions on Neural Networks and Learning Systems*, 27(9):1851–1863, September 2016. doi:10.1109/TNNLS.2015.2458986.

Cedric Fevotte and Jerome Idier. Algorithms for nonnegative matrix factorization with the β-divergence. *Neural Computation*, 23(9):2421–2456, 2011. doi:10.1162/NECO_a_00168.

Christian Walck. *Hand-book on statistical distributions for experimentalists*. 1996. URL `http://www.fysik.su.se/~walck/suf9601.pdf`.

Mehul P Sampat, Mia K Markey, Alan C Bovik, et al. Computer-aided detection and diagnosis in mammography. *Handbook of image and video processing*, 2(1): 1195–1217, 2005.

Peter N Belhumeur, João P Hespanha, and David J Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. Technical report, Yale University New Haven United States, 1997.