Application of Non-Negative Matrix Factorisation Algorithms to Noisy Face Image Data-sets[1]

Abstract

Non-negative matrix factorization (NMF) is one of the effective machine learning technique used for the analysis of high-dimensional data. The algorithm can extract meaningful information even from a sparse non-negative data set. NMF is widely used in image processing as it is more robust to occlusion in images compared to other algorithms.

The objective of this assignment is to implement two Non-Negative Matrix Factorization (NMF) algorithms and analyse the robustness of NMF algorithms when the dataset is contaminated by large magnitude noise by measuring the reconstruction loss.

1 Introduction

Non-negative matrix factorization (NMF) is an unsupervised learning algorithm that can be used to extract sparse and interpretable features. In NMF, we can decompose a non-negative data matrix X as $\boldsymbol{X} \approx \boldsymbol{DR}$ with constraints of $\boldsymbol{D} \geq \mathbf{0}$ and $\boldsymbol{R} \geq 0$, which implies that each data point in $\boldsymbol{D}$ and $\boldsymbol{R}$ are greater than or equal to zero. Given a matrix $\boldsymbol{X}$, NMF can generate two matrices $(\boldsymbol{D}, \boldsymbol{R})$ such that $\boldsymbol{X} \approx \boldsymbol{DR}$. We denote the matrix dimensions as follows. $\boldsymbol{X} \in \mathbb{R}^{(F \times N)}$, $\boldsymbol{D} \in \mathbb{R}^{(F \times K)}$ and $\boldsymbol{R} \in \mathbb{R}^{(K \times N)}$. Note each vector of $\boldsymbol{D}$ can be referred to as basis vectors, and each vector of $\boldsymbol{R}$ can be referred to as weight vectors (further discussed in Section 3).

The purpose of this study is to implement three NMF algorithms and analyse the robustness of NMF algorithms when the dataset is contaminated by large magnitude noise by measuring the reconstruction loss. Implemented algorithms will be applied on two real-world face image datasets, discussed below.

The ORL dataset contains ten different images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open / closed eyes,

[1] Group members: Matthew Martineer (480558396, tutor is Nicholas James); Sajith Ramadasan (460280024, tutor is Baosheng Yu); and Sriram Swaminathan (470495968, tutor is Fengxiang He).

smiling / not smiling) and facial details (glasses / no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement).

The Extended YaleB dataset contains 2414 images of 38 human subjects under 9 poses and 64 illumination conditions. All images are manually aligned, cropped, and resized. The experiment settings include include custom designed noise and applying each algorithm on the both the datasets to analyse the robustness of the algorithm. NMF algorithm from sklearn library will also be used to evaluate and compare the performance of the implemented algorithm with the code that was manually written.

2 Literature Review

The Non-negative Matrix Factorization (NMF) literature is more than thirty years old. The earliest variants of this technique referred to it as non-negative rank factorisation (Chen 1984) or positive matrix factorisation (Paatero and Tapper 1994).

The seminal Lee and Seung (1999) paper used the technique for "learning the parts of objects". They demonstrated the practicality of NMF by expressing image data with a 'sum-by-parts' representation, allowing for the identification and classification of the most important image features (called basis features). This contrasts with vector quantisation and principal components analysis which learn holistic representations. In Lee and Seung (2001), two multiplicative algorithms for NMF were introduced which have become standard in the applied NMF literature.

Following these early studies, numerous NMF algorithms have been developed, and the techniques have been used in a variety of research areas and applications. These include text mining (Berry and Browne 2006), face recognition (Soukup and Bajla 2008), unsupervised object discovery (Sivic et al., 2005), and source processing in speech and music (Virtanen 2007).

More recently, researchers have developed new techniques to ensure NMF is robust to noisy data. Perhaps the most prominent example is Weninger et al. (2008), who use NMF in concurrence with hidden Markov models in the context of speech recognition with the goal of identifying signals within highly noisy and reverberant environments. Bhattacharya et al. (2016) develop an NMF algorithm called the 'heavy NMF model'. If the average noise over large subsets of columns is sufficiently small, they find computational time is less than all existing noisy NMF algorithms that were studied.

3 Methods

C.1 Formulation of NMF Algorithm

NMF involves approximating a data matrix, denoted as $\boldsymbol{X} \in \mathbb{R}^{(F \times N)}$, by factorising it into the product of two matrices, $\boldsymbol{D} \in \mathbb{R}^{(F \times K)}$ and $\boldsymbol{R} \in \mathbb{R}^{(K \times N)}$. $\widehat{\boldsymbol{X}} = \boldsymbol{DR}$ therefore provides an unsupervised linear approximation of $\boldsymbol{X}$, known as the reconstructed matrix. The columns of $\boldsymbol{D}$ are referred to as the 'explanatory variables' or 'patterns' of the original matrix; and the columns of $\boldsymbol{R}$ the 'activation' (or expansion) coefficients. Since data is often non-negative by nature (e.g. pixel intensities, count or price data), the optimal processing of this data will involve non-negative constraints to increase the interpretability of the approximation.

This non-negativity introduces sparsity and part-based decompositions. Specifically, the ith column of $\boldsymbol{X}$ (which contains data for the ith observation) is approximated as $\sum_{k=1}^{K} r_{k,n}\boldsymbol{d}_k$, where $r_{k,n}$ is the (k,n)th element of $\boldsymbol{R}$, and $\boldsymbol{d}_k$ is the dth row of $\boldsymbol{D}$ (the column vector of activation coefficients for the ith observation). This expresses $\boldsymbol{X}$ as the sum of its features, without cancelling each other, which is easily interpretable and intuitive.

The loss function for the NMF approximation is typically in the form $\underset{\boldsymbol{D},\boldsymbol{R} \geq \boldsymbol{0}}{\operatorname{argmin}}\ g(\boldsymbol{X}|\boldsymbol{DR})$, subject to the number of columns of $\boldsymbol{D}$ equalling k, and $g(\bullet \,|\, \bullet)$ is some function measuring the matrix divergence. This produces the solution $g\big(\boldsymbol{X}|\widehat{\boldsymbol{X}}\big)$, where $\widehat{\boldsymbol{X}} = \boldsymbol{DR}$ minimises the objective function. In general, we choose $g\big(\boldsymbol{X}|\widehat{\boldsymbol{X}}\big) = \sum_{f=1}^{F}\sum_{n=1}^{N} \mathrm{s}(x_{fn}|\hat{x}_{fn})$, where $s(\bullet \,|\, \bullet)$ is some continuous non-negative function measuring the *scalar* divergence between x_{fn} and $\hat{x}_{fn}$.

Lee and Seung (1999) employed NMF using two types of scalar divergences: The Euclidean distance, where $\mathrm{s}(x_{fn}|\hat{x}_{fn}) = (x_{fn} - \hat{x}_{fn})^2$; and the Kullback-Leibler (KL) divergence, where $\mathrm{s}(x_{fn}|\hat{x}_{fn}) = x_{fn}ln(x_{fn}/\hat{x}_{fn}) - x_{fn} + \hat{x}_{fn}$. The objective functions can therefore be written as $\|\boldsymbol{X} - \boldsymbol{DR}\|_F^2$ for the Euclidean distance, and $\mathrm{D}(\boldsymbol{X}||\boldsymbol{DR})$ for the KL divergence. Both measures of divergence are convex with respect to both arguments (x_{fn} and $\hat{x}_{fn}$). Observe if both arguments are multiplied by a constant ϑ, the Euclidean distance will increase by ϑ^2, but the KL divergence will increase by ϑ. This implies large values will impact the former substantially more (assuming of course, the divergences are not at a stationary point).

For both these divergences, $g(\boldsymbol{X}|\boldsymbol{DR})$ is convex with respect to $\boldsymbol{D}$ and $\boldsymbol{R}$, but *not* with respect to $\{\boldsymbol{D}, \boldsymbol{R}\}$. Hence, many NMF algorithms rely on an iterative optimisation strategy, where $\boldsymbol{D}$ is updated (with $\boldsymbol{R}$ fixed), and then $\boldsymbol{R}$ is updated (with $\boldsymbol{D}$ fixed). The approach (shown below)

introduced by Lee and Seung (2001; Equations 4 and 5) use the following approaches to optimise the objective function using both the Euclidean distance and the KL divergence. Note $\boldsymbol{A}_{i,j}$ refers to the element in the ith row and jth column of matrix $\boldsymbol{A}$, and initial values of $\boldsymbol{D}$ and $\boldsymbol{R}$ must be chosen before the process begins.

Using the Euclidean distance: $\boldsymbol{R}_{ij} \leftarrow \boldsymbol{R}_{ij} \frac{(\boldsymbol{D^T X})_{ij}}{(\boldsymbol{D^T D R})_{ij}}$; $\boldsymbol{D}_{ij} \leftarrow \boldsymbol{D}_{ij} \frac{(\boldsymbol{X R^T})_{ij}}{(\boldsymbol{D R R^T})_{ij}}$

Using the KL divergence: $\boldsymbol{R}_{ij} \leftarrow \boldsymbol{R}_{ij} \frac{\sum_k \boldsymbol{R}_{jk} \boldsymbol{X}_{ik} / (\boldsymbol{DR})_{ik}}{\sum_l \boldsymbol{H}_{jl}}$; $\boldsymbol{D}_{ij} \leftarrow \boldsymbol{D}_{ij} \frac{\sum_k \boldsymbol{R}_{jk} \boldsymbol{X}_{ik} / (\boldsymbol{DR})_{ik}}{\sum_l \boldsymbol{R}_{jl}}$

This process ensures the nonnegativity of $\boldsymbol{D}$ and $\boldsymbol{R}$ is guaranteed by construction.

An alternate to the Lee and Seung (2001) approach is the Hierarchical Alternating Least Squares (HALS) algorithm introduced by Cichocki and Phan (2009). Defining $\boldsymbol{X}^j = \boldsymbol{X} - \sum_{k \neq j} \boldsymbol{d}_j \boldsymbol{r}_j$ as the residual without the jth column ($\boldsymbol{d}_j$ is the jth column of $\boldsymbol{D}$, $\boldsymbol{r}_j$ the jth row of $\boldsymbol{R}$), they introduce a 'vector-wise' objective function of $\left\|\boldsymbol{X}^j - \boldsymbol{d}_j \boldsymbol{r}_j\right\|_F^2$. Setting the gradient of the function equal to 0 produces the closed-form optimal solution $\boldsymbol{d}_j^* = max(\boldsymbol{X}^j \boldsymbol{d}_j / \boldsymbol{r}_j \boldsymbol{r}_j^{\boldsymbol{T}}, 0)$. After making the normalisation $\left\|\boldsymbol{d}_j\right\|_2 = 1$ to avoid the scalar in the denominator, the update rule of each vector of $\boldsymbol{D}$ and $\boldsymbol{R}^T$ is as follows. The update rule must be done for all $j = 1, \ldots, K$:

$$\boldsymbol{d}_j \leftarrow \max(\boldsymbol{X}^j \boldsymbol{r}_j^T, 0); \qquad \boldsymbol{r}_j^T \leftarrow \max(\boldsymbol{X}^{j^T} \boldsymbol{d}_j, 0)$$

The algorithm converges faster than the Lee and Seung (2001) multiplicative update rules since each vector-wise update achieves an optimal (closed-form) solution.

The number of matrices of rank-1 matrices within the approximation is known as k. This is equivalent to the number of columns in the dictionary matrix, $\boldsymbol{D}$; or the number of rows in the reconstruction matrix, $\boldsymbol{R}$. When NMF is applied to a matrix of vector images, it can be interpreted as the number of facial features. This means the columns of $\boldsymbol{D}$ each represent a certain facial feature, and the columns of $\boldsymbol{R}$ show the importance of the features in each image.

An appropriate choice of k is critically important for NMF. Whilst a greater k results in a more accurate reconstruction of the data matrix $\boldsymbol{X}$; a smaller k results in a more parsimonious model, reducing computation time and increasing interpretability. The value chosen will depend on the data matrix as well as the application. Results for both a smaller value, and a larger value, of k is presented to trade-off these competing factors. These values are $k = 2$ as the smaller value; and $k = 10$ for the ORL data-set, and $k = 15$ for the Yale B data-set.

C.2 Noise choice and description

Denote $\boldsymbol{X_1} \in \mathbb{Z}^{(2576\times400)}$ as the matrix of vectorised 8-bit images for the ORL data, and $\boldsymbol{X_2} \in \mathbb{Z}^{(2016\times2414)}$ as the matrix of vectorised images for the Yale B data. Note $\boldsymbol{X_1}^{[i,j]}, \boldsymbol{X_2}^{[i,j]} \in \{0,\ldots,255\}\ \forall i,j$; representing the 256 possible shades of grey.

Three different types of noise have been analysed for this report. The noise for the ORL data is represented by $\boldsymbol{W_1^p} \in \mathbb{R}^{(2576\times400)}$ $(p = 1,2,3)$. This is a random matrix, where each element is independently and identically distributed in accordance with some distribution. If $p = 1$, we use a normal distribution; if $p = 2$, we use a uniform distribution (the maximum entropy probability distribution); and if $p = 3$, we use a salt-and-pepper distribution. The noise for the Yale B data, $\boldsymbol{W}_2^p \in \mathbb{R}^{(2016\times2414)}$, is also represented in a random matrix, defined similarly. We therefore generate six different 'noise matrices' – using the three different types of noise across the three data-sets.

The parameters for the three distributions were chosen as follows. For the Gaussian noise $(p = 1)$, we assume each element in the random matrix is Gaussian distributed with 0 mean and a standard deviation of 20, which leads to substantial contamination of the images. For the uniform noise, we choose a symmetric distribution centred around 0, implying the noise can be expressed as a $\mathrm{U}[-\delta,\delta]$ distribution. To directly compare the impact of Gaussian vs. uniform noise on the NMF result, we choose δ such that the variance of the normally distributed noise equals the variance of the uniformly distributed noise. (This prevents an incorrect interpretation – believing that one noise is more robust than another, when it's due to differing variance, rather than different *noise types*). Since the variance of a $\mathrm{U}[-\delta,\delta]$ random variable is $\frac{1}{12}(2\delta)^2$, we equate this with 20^2 (the variance of the Gaussian noise) which yields a value of $\delta = \sqrt{3\times20^2} = 35$ (rounded to the nearest integer). Finally, our third type of noise, salt-and-pepper noise was chosen since it's a type of noise often seen in images. Salt-and-pepper noise is a noise having white and black pixels sparsely present on an image. Using a random generator function, a set of random co-ordinates are generated, and the pixel values is set to white or black colour. Thus, generated noise is added to the original image to create a noisy image.

A corrupted matrix of vectorised images (for the pth noise type) for the ORL data is equal to $|\boldsymbol{X_1} + \boldsymbol{W_1^p}| \in \mathbb{Z}^{(2576\times400)}$, where the $|\bullet|$ operator does the following: (a) If any element of the matrix exceeds 255, set the value of the element to 255; (b) If any element is negative, set the value of the element to 0; and (c) Rounds each element to the nearest integer. This ensures *all* elements of the corrupted matrix $\in \{0,\ldots,255\}$, allowing transformation back to an 8-bit image. Similarly, the corrupted matrix for the Yale B data is to $|\boldsymbol{X_1} + \boldsymbol{W_1^p}| \in \mathbb{Z}^{(2016\times2414)}$.

We now have six different corrupted data matrices. Let d denote the data-set ($d = 1$ refers to the ORL data; $d = 2$ to the Yale B data). Let $\boldsymbol{X}^{(d,p)}$ be the dth data-set ($d = 1,2$) corrupted by the pth noise type ($p = 1,2,3$). We implement a selection of NMF algorithms to the six $\boldsymbol{X}^{(d,p)}$ matrices in the next section, and examine their robustness to each noise type.

4 Experiments and Discussion

Three separate NMF algorithms are implemented in this report. These are:

- The Lee and Seung (2001) multiplicative update rules using the Euclidean distance;
- The Lee and Seung (2001) multiplicative update rules using the Kullback-Leibler divergence;
- The Cichocki and Phan (2009) Hierarchical Alternating Least Squares (HALS) algorithm.

The results for when the algorithms are applied to the ORL data are presented below. We include the results from Sklearn (NMF from SKLearn for evaluating our algorithms), which uses default parameters with no optimisation.

Algorithm	Noise	K	RRE	Accuracy	NMI
Multiplicative update	Gaussian Noise	2	0.274599198	0.4	0.6065
		10	0.237465661	0.555	0.7384
		40	0.30463495	0.1825	0.3513
	Salt Pepper Noise	2	0.280535366	0.395	0.6054
		10	0.255447288	0.4825	0.6635
		40	0.308863394	0.1775	0.3291
	Random Noise	2	0.272822562	0.4125	0.6132
		10	0.227898159	0.575	0.7574
		40	0.303601531	0.185	0.3409
skLearn	Gaussian Noise	2	0.274574495	0.4075	0.63
		10	0.216440314	0.64	0.798
		40	0.201860868	0.66	0.8028
	Salt Pepper Noise	2	0.280386829	0.43	0.6309
		10	0.230400865	0.6225	0.776
		40	0.239426931	0.565	0.7179
	Random Noise	2	0.27277852	0.4275	0.6422
		10	0.20762962	0.6625	0.8026
		40	0.163049641	0.69	0.8308
Hierarchal Alternating Least Squares (HALS)	Gaussian Noise	2	0.280209741	0.3325	0.5699
		10	0.249594982	0.4575	0.6866
		40	0.221343228	0.6375	0.7988
	Salt Pepper Noise	2	0.286556594	0.355	0.5885
		10	0.262603811	0.4525	0.6551
		40	0.247450665	0.535	0.7179
	Random Noise	2	0.278252601	0.3575	0.5963
		10	0.243063734	0.495	0.7011
		40	0.203379737	0.685	0.8275

The results for when the algorithms are applied to the Yale B data are presented below.

Algorithm	Noise	K	RRE	Accuracy	NMI
Multiplicative update	Gaussian Noise	2	0.385927707	0.0824	0.0818
		10	0.546529232	0.0853	0.0843
		40	0.544463284	0.0849	0.0863
	Salt Pepper Noise	2	0.398230083	0.0795	0.0827
		10	0.553492926	0.0824	0.0852
		40	0.551365365	0.082	0.0867
	Random Noise	2	0.374368271	0.0837	0.0891
		10	0.539579737	0.0841	0.0836
		40	0.537279798	0.0862	0.0918
skLearn	Gaussian Noise	2	0.383816219	0.0837	0.0852
		10	0.283566291	0.1114	0.1346
		40	0.228198118	0.2154	0.305
	Salt Pepper Noise	2	0.396066197	0.0808	0.0859
		10	0.30313603	0.1106	0.1128
		40	0.261009968	0.2291	0.3235
	Random Noise	2	0.373326791	0.0845	0.0893
		10	0.265484002	0.1156	0.1341
		40	0.198107398	0.21	0.2832
Hierarchal Alternating Least Squares (HALS)	Gaussian Noise	2	0.438139431	0.0829	0.0894
		10	0.494317508	0.0825	0.0767
		40	0.492251561	0.0795	0.0787
	Salt Pepper Noise	2	0.346018359	0.0824	0.0751
		10	0.501281203	0.082	0.0776
		40	0.499153642	0.0837	0.0791
	Random Noise	2	0.322156548	0.0841	0.0815
		10	0.487368014	0.0862	0.076
		40	0.485068074	0.0837	0.0842

HALS takes comparatively more time to converge for YaleB dataset compared to ORL dataset as K increases. This might be due to more columns in YaleB dataset and HALS involves processing each column. The two Lee and Seung (2001) algorithms gave very similar results, so only the KL divergence is presented in the table.

Below, the RRE is shown for a variety of K and noise types, across both data-sets.

Image Data	K	Noise	Algorithm	RRE
ORL	2	Uniform	L2-Norm	0.31
ORL	15	Uniform	L2-Norm	0.25
ORL	40	Uniform	L2-Norm	0.22
ORL	2	Uniform	KL Divergence	0.31
ORL	15	Uniform	KL Divergence	0.25
ORL	40	Uniform	KL Divergence	0.22
ORL	2	Gaussian	L2-Norm	0.31
ORL	15	Gaussian	L2-Norm	0.31
ORL	40	Gaussian	L2-Norm	0.31
ORL	2	Gaussian	KL Divergence	0.31
ORL	15	Gaussian	KL Divergence	0.31
ORL	40	Gaussian	KL Divergence	0.31
ORL	2	Salt & Pepper	L2-Norm	0.31
ORL	15	Salt & Pepper	L2-Norm	0.31
ORL	40	Salt & Pepper	L2-Norm	0.31
ORL	2	Salt & Pepper	KL Divergence	0.31
ORL	15	Salt & Pepper	KL Divergence	0.31
ORL	40	Salt & Pepper	KL Divergence	0.31
YaleB	2	Uniform	L2-Norm	0.43
YaleB	15	Uniform	L2-Norm	0.39
YaleB	38	Uniform	L2-Norm	0.37
YaleB	2	Uniform	KL Divergence	0.43
YaleB	15	Uniform	KL Divergence	0.38
YaleB	38	Uniform	KL Divergence	0.36
YaleB	2	Gaussian	L2-Norm	0.35
YaleB	15	Gaussian	L2-Norm	0.35
YaleB	38	Gaussian	L2-Norm	0.35
YaleB	2	Gaussian	KL Divergence	0.35
YaleB	15	Gaussian	KL Divergence	0.35
YaleB	38	Gaussian	KL Divergence	0.35
YaleB	2	Salt & Pepper	L2-Norm	0.35
YaleB	15	Salt & Pepper	L2-Norm	0.35
YaleB	38	Salt & Pepper	L2-Norm	0.35
YaleB	2	Salt & Pepper	KL Divergence	0.35
YaleB	15	Salt & Pepper	KL Divergence	0.35
YaleB	38	Salt & Pepper	KL Divergence	0.35

Usually, as the value of K to define the weight matrix goes high and as close to the image column size, the KL divergence NMF shows a much better performance than L2-Norm NMF. Thus, larger value of K does have an impact in the lower reconstruction loss.

Relative Reconstruction Errors (RRE's) are used to understand the image reconstruction for both the image datasets- ORL and Cropped YaleB against different values of K and noise contaminations. It is observed that reconstruction of the ORL images are much better than the reconstruction for the Cropped YaleB.

It is also observed that images contaminated with Salt & Pepper noise are much easier to reconstruct. With these contaminated images, both the NMF algorithms with L2-Norm and KL Divergence technique against any value of K gives the same result.

Figure 1 is shown on the next page, which shows the impact of noise on the both the original and reconstructed images. Figure 2 is shown on page 11, which is a plot of the loss functions. It is observed that saturation is attained at about 100 iterations and both the L2-Norm and KL Divergence almost have the same loss and error. Analysis of the individual facial features is shown in Appendix B, and a further exploration of the HALS algorithm in shown in Appendix C.

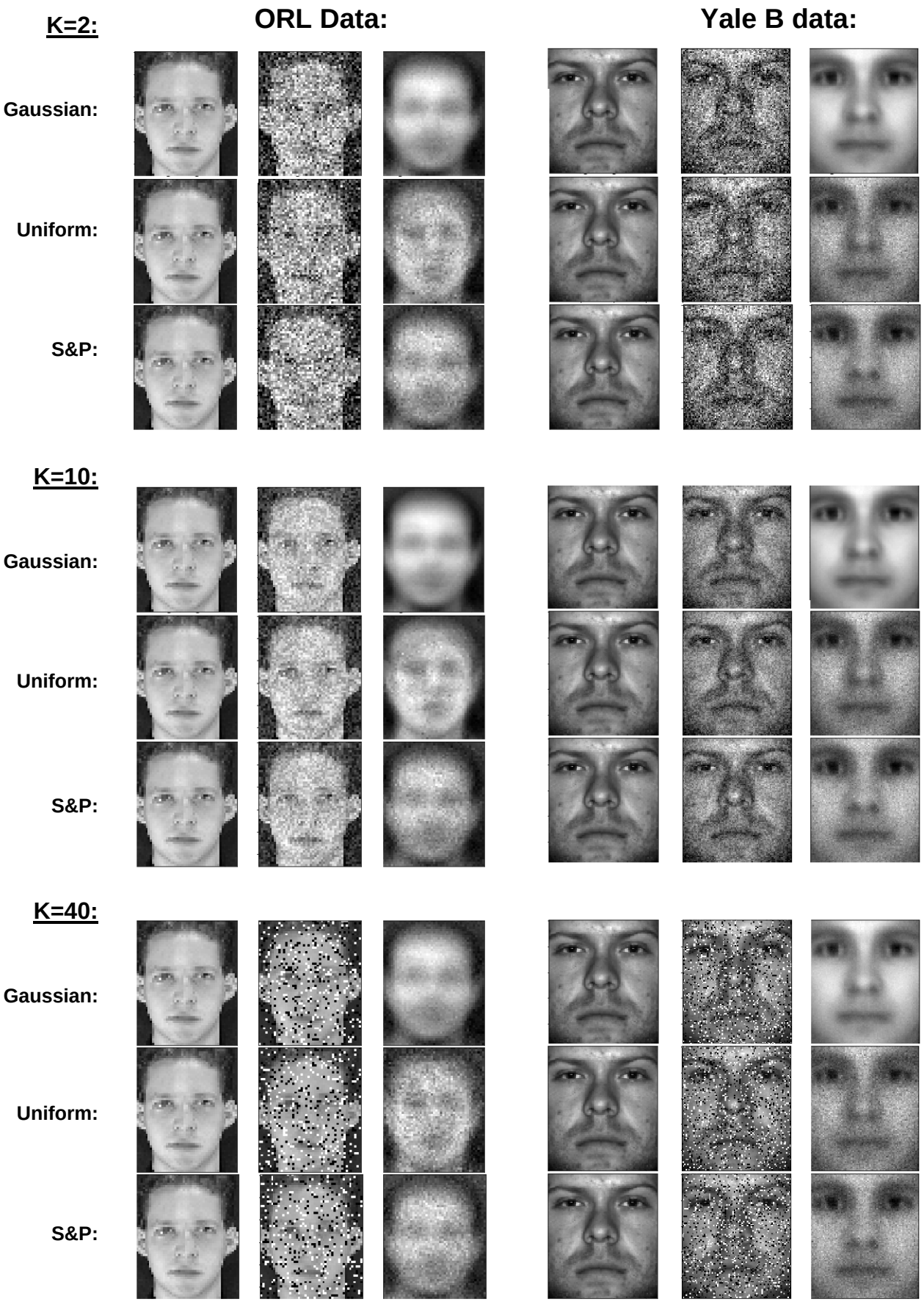

Figure 1: An arbitrary image from the ORL data-set (left panel) and Yale B data-set (right panel) with three different values for *K* and three different types of noise (as labelled). The left and right panels consist of 9 rows, showing the original image (left), the image corrupted by noise (middle), and the reconstructed image (right) – reconstructed using the Lee and Seung (2001) algorithm with KL divergence.

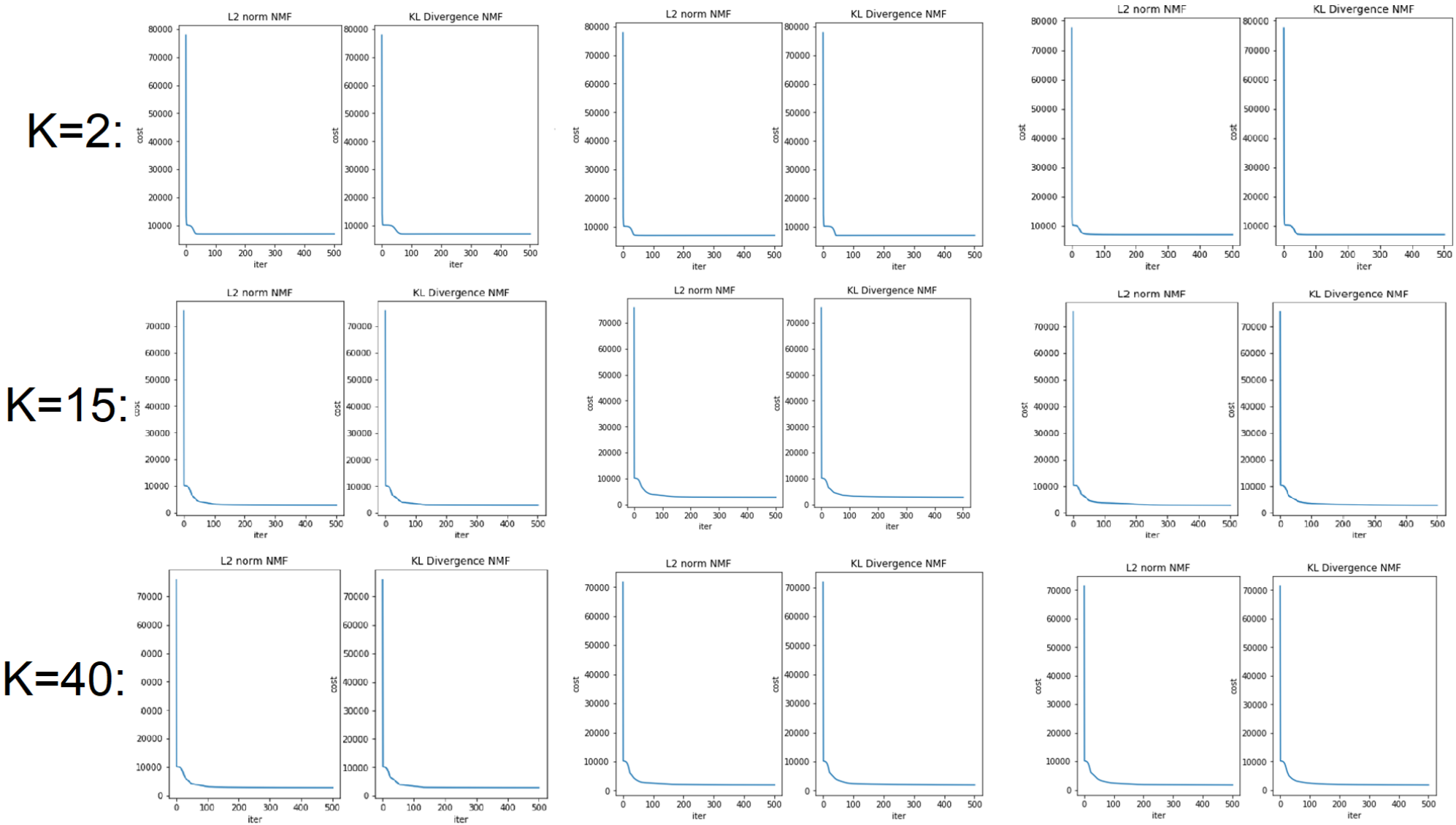

Figure 2: Convergence analysis (showing loss functions of the ORL data) of the Lee and Seung (2001) algorithm, using the L2 Norm (Euclidean distance) and KL divergence for three *K* values, and for the three noise types. The algorithm steps follow: (1) The image data set is loaded and are added with three different types of noise, namely-uniform random noise, Gaussian noise and salt & pepper noise. (2) The NMF Algorithms with Multiplicative update for D and R using L2-Norm objective and KL-Divergence objective functions are used. (3) The image loaded calls for the NMF function defined for L2 Norm and KL separately. (4) Three different k-values are tried using three different noise, namely-uniform distribution, Gaussian and salt & pepper noise. (5) Both the algorithm runs to convergence when it reaches the number of iterations or when the delta between the 'W' is less than the tolerance (<e5). (6) Performance Evaluations are done by calculating the RRE (Relative Reconstruction Error) based on the contaminated images.

5 Conclusion

This report has shown how three different types of noise (Gaussian, uniform, salt-and-pepper) and the choice of the parameter K (the number of features used to reconstruct the original matrix) impacts three different types of NMF algorithms applied to face image data. These algorithms were the Lee and Seung (2001) multiplicative update rule (using both the Euclidean distance and the KL divergence), and Cichocki A and Phan's (2009) Hierarchal Alternating Least Squares (HALS) algorithm. In terms of future work, the optimisation of HALS algorithm to improve the performance could be further explored. The use of additional noise types or evaluation metrics will also be useful to determine the optimal NMF's to apply for face image data.

The contribution of group members follows. Matthew (480558396) wrote all sections of the report (excluding Sriram's contribution) and wrote some of the Lee and Seung (2001) algorithms (e.g. introducing noise and determining the parameters of the noise). Sajith (460280024) wrote the rest of the Lee and Seung (2001) algorithms (e.g. multiplicative updates), the Cichocki and Phan (2009) algorithm, and produced the images that were used in Figure 1. Sriram (470495968) produced the plots that were used in Figure 2 and wrote the Abstract, Appendix A, and some of the introduction.

References

Bhattacharya C, Goyal N, Kannan R & Pani, J. Non-negative matrix factorization under heavy noise. Proceedings of The 33rd International Conference on Machine Learning. 2016;48:1426-1434.

Berry M and Browne M. Email surveillance using nonnegative matrix factorization. Computational & Mathematical Organization Theory. 2005;11:249–264.

Chen J. The nonnegative rank factorizations of nonnegative matrices. Linear Algebra and its Applications. 1984;62:207-217.

Cichocki A and Phan A. Fast local algorithms for large scale nonnegative matrix and tensor - factorizations. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. 2009;92:708-721.

Kimura K. A study on efficient algorithms for nonnegative matrix/tensor factorization. PhD Dissertation, Hokkaido University. 2017.

Lee D and Seung H. Learning the parts of objects by non-negative matrix factorization. Nature. 1999;401:788-791.

Lee D and Seung H. Algorithms for non-negative matrix factorization. In Advances in Neural Information Processing. 2001;13

Paatero P and Tapper U. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. Environmetrics. 1994;5:111-126.

Sivic J, Russell B, Efros A, Zisserman A & Freeman W. Discovering object categories in image collections. In Proceedings of the international conference on computer vision. 2005;22:198-214.

Soukup D and Bajla I. Robust object recognition under partial occlusions using NMF. Computational Intelligence and Neuroscience. 2008;4:64-78.

Virtanen T. Monaural sound source separation by non-negative matrix factorization with temporal continuity and sparseness criteria. IEEE Transactions on Audio, Speech and Language Processing. 2007;15:1066–1074.

Weninger F, Wollmer M, Geiger J, Schuller B, Gemmeke J, Hurmalainen A, Virtanen V & Rigoll G. Non-negative matrix factorization for highly noise-robust ASR: To enhance or to recognize? IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, 2012;4681-4684.

Appendix A (Procedure to Run Code)

The Code will run in Anaconda environment with proper dependencies. Environment used for development is Python 3.6.

For a full list of packages used, please see the code. There are two ipython notebooks that are broken out into different parts of the assignment. These notebooks can be loaded in a jupyter environment and cells executed sequentially. The procedure to run the code is as follows.

1. Change the file location for the images to load the image data file at the forth and tenth code blocks where we load ORL and Cropped YaleB respectively.
2. The codes are organised in such as way that ORL database is first loaded and the different NMF algorithms are then run.
3. Once, the cost of the reconstruction and the reconstruction errors are printed for the ORL, the subsequent codes are then used to run Cropped YaleB image dataset.
4. Please, note that cropped YaleB Image takes more time (~30 mins) to run the NMF algorithms (can be reduced by setting K=2).
5. The graphs are plotted for the evaluation of the loss for the reconstructed images along with print out of relative reconstruction errors.
6. Please, note that the code contains three different types of noise and three different K values to evaluate performance of the NMF Algorithms.

Appendix B (Analysis of Facial Features)

A selection of images from the ORL data-set (left) and Yale B data-set (right) is shown below.

Four panels of data follow. Each panels consists of the feature set (columns of $\boldsymbol{D}$), made up of $K = 10$ features; followed by a 5×10 matrix containing the first 50 reconstructed images using:

- *Panel A:* Lee and Seung's (2001) algorithm; Gaussian noise; $K = 10$; ORL data-set.
- *Panel B*: Same as above, but for the Yale B data-set
- *Panel C*: Cichocki and Phan's (2009) HALS algorithm; Gaussian noise; $K = 10$; ORL data-set.
- *Panel D*: Same as above, but for the Yale B data-set.

Panel A:

Panel B:

Panel C:

Panel D:

Appendix C (Further Analysis of HALS algorithm)

A convergence analysis of the HALS algorithm is shown below, for the ORL (left panel) and Yale B data-set (right panel).