COMP5328 - Assignment 1 - Non-negative Matrix Factorization

Tutors: Jeremy Gillen, Yu Yao
Group members: Kilian Liss (500412817) - Alexander Mars (450347582) - James Wood (480344386)

Abstract

To represent large data sets into smaller components, NMF algorithms were implemented based on 4 different loss functions: L_2-norm, Regularized L_2-norm, $L_{2,1}$-norm, and a $tanh$-based loss, and a discussion on various other NMF algorithms is provided. Each algorithm was tested on the ORL and YaleB data set of faces, with varying amounts of noise for noise types of random white pixels, random white boxes and Laplacian noise. Their performance were tested and analysed using the evaluation metrics of Relative Reconstruction Error (RRE), Accuracy and Normalized Mutual Information (NMI). We found that Regularized L_2-norm NMF consistently achieved the lowest RRE across different noise types and values.

1 Introduction

Advances in technology have lead to massive amounts of raw data being generated across a variety of industries. Such data is not only produced in research domains, but also by businesses and individuals in day to day life. A security camera for example, captures gigabytes of image data on a daily basis. The current widespread generation and use of large complex data has spurred a requirement for efficient methods that can extract meaningful information from such data.

One approach is to transform the data into a matrix, and representing the matrix in a lower-dimensional state by means of factorization, and maximising the information retained [1]. In this context, the purpose of factorisation is to compress the data, and identify the key components of the original data that are maximally informative.

Dictionary learning is an area of machine learning which tries to learn a factorisation of a data matrix into two smaller matrices: a dictionary, and a set of coefficients. The dictionary matrix represents the set of common components of the data, and is analogous to a set of basis vectors, whilst the coefficients weight the contribution of each component.

Formally, dictionary learning attempts to represent the original data matrix $R \in \mathbb{R}^{n \times d}$ in terms of left factors $P \in \mathbb{R}^{n \times k}$ and right factors $Q \in \mathbb{R}^{k \times d}$ such that $R \approx PQ$. Here, we say that the matrix R has n entries (rows) with d features for each entry (columns), and k is the number of latent variables used to reconstruct R. In this way, R is factorized into $k << (d \,\&\, N)$ components, which can significantly compress the data. Learning P and Q is achieved by minimising a loss function on the reconstruction, where the loss function f is generally of the form $f(R - PQ)$ as it attempts to measure the difference between the original data and the learned representation.

Non-negative Matrix Factorization (NMF) is a special kind of dictionary learning, in which the matrices R, P and Q are constrained to be strictly non-negative. Introducing a non-negativity constraint makes NMF applicable to real-world data, which is often inherently positive. For example, audio

signals, genetic signals, and astronomy data are inherently non-negative [reference]. In addition, non-negative representations of data are easier to interpret and visualise.

Since noise is often present in real world data, an NMF must also be robust to noise in order to be used for practical purposes. Here, robustness is defined as the ability of the algorithm to learn a good representation for R (learn P and Q) despite the presence of noise. In some cases, it is said that $R = QP + N$ where N is the error on the reconstruction, and may be assumed to be due to noise. Different NMF algorithms have been designed by minimizing different loss functions, which is suited to different distributions from which the noise N may be drawn.

In this paper, we first describe some robust NMF algorithms including their strengths and weaknesses. We then describe two face image data sets; the ORL and Yale data sets, three different methods of introducing noise to these data sets, and metrics for evaluating the robustness of NMF algorithms. Finally, we examine the robustness of four different NMF algorithms in response to varying levels of noise. Our paper may serve as a benchmark for other researchers to compare the performance of novel NMF algorithms. We also introduce a simple regularisation method for the classic L_2-NMF which we believe to be our original idea that yields surprisingly good results.

2 Related Work

2.1 Alternative Robust NMF Algorithms

Here we review a selection of NMF algorithms that aim to be robust to noise, but were not implemented in this paper due to computational constraints. For a description of the algorithms implemented in this paper, we refer the reader to section 3.2 in which we explain the NMF algorithms implemented in detail.

2.1.1 L_1-norm NMF

The objective function for the L_1-NMF is based on the L_1-norm which for a matrix A is defined as $||A||_1 = \sum_{ij} |A_{ij}|$ where $|\cdot|$ is the absolute value function. The optimisation problem is then defined as $\min_{Q,P} ||R - QP||_{2,1}$ s.t. $Q \geq 0, P \geq 0$. Trying to derive a solution based on gradient descent as in the case of L_2-norm (section 3.2.1) fails in the context of the L_1-norm as it requires the inverse of a non-invertible matrix. Lam [2] proposed a solution to this problem under the assumption that noise in the data was additive and drawn from a Laplacian distribution. The solution is presented in the form of a linear programming algorithm, which was not in the scope of this work to implement. However, L_1-NMF has been shown to be more robust to noise than L_2-NMF, which can be understood intuitively by comparing the L_2 and L_1 norms. The L_2-norm squares the error on $R - PQ$, and thus is more susceptible to large outliers.

2.1.2 L_1 Regularised NMF

An alternative approach to L_1-NMF assumes that noise in the data can have arbitrarily large values, but that the noise is sparsely distributed [3]. This is formulated as $R = QP + S$ where $S \in \mathbb{R}^{n \times d}$ is an error matrix to explicitly capture sparse corruption. The optimisation problem is formulated as $\min_{Q,P,S} ||R - QP - S||_F^2 + \lambda ||S||_1$ s.t. $Q \geq 0, P \geq 0, \lambda > 0$ where $||\cdot||$ is the L_1-norm as specified in 2.1.1. Here λ is the regularisation parameter that controls the sparsity of S. The algorithm for optimising L_1 regularised NMF is based on quadratic programming and determining the appropriate value for λ is difficult, thus we did not implement it in this paper.

2.1.3 Hypersurface Cost Based NMF

Hypersurface Cost based NMF employs an objective function based on the hypersurface cost function (HCF) first introduced in [4]. The HCF function p is defined as $p(||R - QP||) = \sqrt{1 + ||R - QP||^2} - 1$, and is proposed to be robust based on it being quadratic when $||R - QP||$ is small, and linear when $||R - QP||$ is large. Furthermore, the influence function of HCF, that is, the function that defines its response to infinitesimally small changes in the input, is differentiable

[5]. Subsequently, by minimising the HCF function with non-negativity constraints on Q and P, [5] show that the following update rules can be yielded for P and Q:

$$\begin{aligned} Q' &= Q - \alpha \frac{QPP^T - RP^T}{\sqrt{1 + ||R - QP||}} \\ P' &= P - \beta \frac{Q^T QP - Q^T R}{\sqrt{1 + ||R - QP||}} \end{aligned} \tag{1}$$

where β and α are the step sizes and must be determined by an Armijo line search [6] at each iteration. The line search makes hypersurface cost based NMF a time consuming algorithm to run[5], thus we did not implement it in this paper.

2.1.4 Correntropy Induced Metric NMF

In similar fashion to Hypersurface Cost Based NMF, Correntropy Induced Metric NMF (CIM-NMF) uses an alternative objective function based on the CIM in order to improve robustness. The CIM for two vectors x, and y is defined as $CIM(x, y) = (k(0) - \frac{1}{n}\sum_{i=1}^{n} k_\sigma(e_i)^{1/2}$ where $e_i = x_i - y_i$ and k is the kernel function [7]. The kernel used for CIM-NMF is the Gaussian kernel given by $g(e, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} exp(-e^2/2\sigma^2)$. This means the CIM has a value close to 1 when the error value e is large, therefore improving the robustness of CIM to large outliers. Du et al. [8] show that the objective function for CIM-NMF can then be defined as

$$J(Q, P) = 1 - \frac{1}{nd}\sum_{i=1}^{n}\sum_{j=1}^{m} g(R_{ij} - \sum_{k=1}^{k} Q_{ik}P_{jk}, \sigma),$$

which is equivalent to solving the optimisation problem

$$\min_{Q,P} \sum_{i=1}^{n}\sum_{j=1}^{m} g(R_{ij} - \sum_{k=1}^{k} Q_{ik}P_{jk}, \sigma) \text{ s.t. } Q \geq 0, P \geq 0.$$

The solution is given by a half-quadratic programming algorithm, which was out of the scope of this paper to implement. Du et al. also propose rCIM-NMF which considers entries within a single row as unified, therefore aiming to eliminate the influence of outliers in a row-wise manner.

2.1.5 Truncated Cauchy NMF

More recently, Guan et al. [9] proposed an NMF algorithm that cut-off large errors using a truncation parameter. The objective function is based on Cauchy loss given by $l(x, \gamma) = \ln(1 + (x/\gamma)^2)$, where γ is the scale parameter of the Cauchy distribution. The Cauchy loss function is less sensitive to outliers, but cannot handle infinitely large noise. To limit the effect of large noise, Guan et al. propose using a cutoff σ, such to create a Truncated Cauchy objective function for use with NMF, which is defined as:

$$\min_{Q,P} \frac{1}{2}\sum_{ij} g((\frac{R - QP}{\gamma})_{ij}^2),$$

With:

$$g(x) = \begin{cases} ln(1 + x), 0 \leq x \leq \sigma \\ ln(1 + \sigma), x \geq \sigma \end{cases} .$$

They show that Truncated Cauchy is better able to handle both moderate and large outliers, on multiple types of noise, however it requires optimisation via a half-quadratic programming algorithm, along with optimisation of the γ and σ parameters.

2.2 Similar Studies

Guan et al. [9] conducted a recent comparison of the robustness of multiple NMF algorithms in response to noise. They found that their novel approach of using Truncated Cauchy Loss had improved robustness upon all NMF algorithms mentioned in this paper, except for tanh-NMF [10] which was

not examined. In [10], Shen et al. compare the performance of $tanh$NMF to a number of algorithms, although they do not include the L_2-regularised NMF algorithm employed in this paper, and did not utilise the ORL data set. Furthermore, their analysis focused on accuracy and normalised mutual information rather than relative reconstruction error, whereas we utilise all three in our experimental approach.

3 Methods

For instructions on how to run the code associated with this paper, refer to appendix A. For contributions, see appendix C. Here we cover the data used, the four NMF algorithms implemented, and the three noise algorithms used.

3.1 Data and Preprocessing

To test the performance of our NMF algorithms, we used two data sets. In the YaleB data set [13], each face has a neutral facial expression, the faces are manually centered and re-sized, but the lighting conditions vary between each image . In the ORL data set [14], the lighting condition remain the same, but the facial expressions vary, and the faces are not necessarily aligned. A representative sample of both data sets are shown in figure 1. The YaleB data set contains 2414 images of size 192×168, and the ORL set has 400 images of size 112×92. To reduce computational complexity, we reduced the resolution of each image.

Figure 1: Comparing the YaleB and ORL data set

3.2 NMF Algorithms

We examined four different NMF algorithms in this paper, classical L_2-norm NMF as a benchmark for comparison, a regularised version of L_2-norm NMF, the $L_{2,1}$-norm NMF, and $tanh$NMF [10]. In the following subsections we give the details of each of these algorithms, and comment on their expected robustness to noise.

3.2.1 L_2 Norm NMF

The L_2-norm based NMF was first introduced in [11], we reproduce the details of the derivation here to demonstrate how an NMF algorithm may be derived. The general process of deriving an NMF algorithm involves specifying a loss function that measures the difference between the original data

R and the reconstructed data PQ^T. We then derive a process to minimize the loss function with respect to P and Q, with the condition that neither matrices contains negative elements.

We can measure the reconstruction error using an L_2-norm loss function as follows:

$$l(R, P, Q) = ||R - PQ^T||_2^2 \tag{2}$$

We can minimize Eq. (2) by iteratively moving down in the direction of the gradient of $l(R, P, Q)$ with respect to P, then move down its gradient with respect to Q, and repeat until some stopping criteria is met. In other words, using a step size of α, we update P using:

$$P' = P - \alpha \frac{dl}{dP} \tag{3}$$

then update Q using:

$$Q' = Q - \alpha \frac{dl}{dQ} \tag{4}$$

and repeat until convergence. To ensure that our factors P and Q remains strictly positive we calculate an appropriate step size α at each iteration, which we will derive below.

Update Rule for P: Using the L_2-norm loss function, Eq. (3) becomes:

$$P' = P - \alpha PQ^TQ + \alpha RQ \tag{5}$$

So to avoid any negative terms, we choose an α such that:

$$\begin{aligned} P - \alpha PQ^TQ &= 0 \\ \alpha &= \frac{P}{PQ^TQ} \end{aligned} \tag{6}$$

And so our update rule for P becomes:

$$P' = \alpha RQ = P\frac{RQ}{PQ^TQ} \tag{7}$$

Update Rule for Q: Using the same methodology as for P, we find that the required update rule for Q becomes:

$$Q' = Q\frac{R^TP}{QP^TP} \tag{8}$$

3.2.2 L_2 Norm With Regularizer

After adding a regularizing term β into our L_2-norm loss function, it becomes:

$$l(R, P, Q) = ||R - PQ^T||_2^2 + \beta(\|P\|_2^2 + \|Q\|_2^2) \tag{9}$$

This additional term will penalize any terms in P or Q that become too large, which should reduce over-fitting. Putting this new loss function into our generalized gradient descent update rule from Eq. (3) we get:

$$P' = P - \alpha PQ^TQ + \alpha RQ - \alpha\beta P \tag{10}$$

To avoid the negative terms, we chose α such that:

$$\begin{aligned} P - \alpha PQ^TQ - \alpha\beta P &= 0 \\ \alpha &= \frac{P}{PQ^TQ + \beta P} \end{aligned} \tag{11}$$

and so our update rule for P becomes:

$$P' = \alpha RQ = P\frac{RQ}{PQ^TQ + \beta P} \tag{12}$$

Using a similar derivation for Q, our update rule for Q becomes:

$$Q' = Q\frac{R^TP}{QP^TP + \beta Q} \tag{13}$$

3.2.3 $L_{2,1}$ Norm Algorithm

We implemented the algorithm for $L_{2,1}$ NMF, as described by Kong et al. [12]. This algorithm uses the $L_{2,1}$-norm as the loss function, and is proposed to be robust to Laplacian noise. The objective function for $L_{2,1}$ NMF is defined as

$$||R - QP||_{2,1} = \sum_{i=1}^{n}\sqrt{\sum_{j=1}^{p}(R - QP)_{ji}^2} = \sum_{i=1}^{n}||r_i - Qp_i||_2.$$

Here, the error is computed as a sum of unsquared 2-norms with respect to each feature (the $||r_i - Qp_i||_2$ terms), which reduces the influence of larger errors in comparison to the original Frobenius norm of the L_2 loss function (with or without regularisation).

The optimisation problem can be summarised as

$$\min_{Q,P} ||R - QP||_{2,1} \quad \text{s.t. } Q \geq 0, P \geq 0.$$

The update rules are then given by

$$Q' = Q\frac{(RDP^T)}{(QPDP^T)}$$

$$P' = P\frac{(Q^TRD)}{(Q^TQPD)}$$

where D is a diagonal matrix with the diagonal elements given by

$$D_{ii} = \frac{1}{\sqrt{\sum_{j=1}^{p}(R - QP)_{ji}^2}} = \frac{1}{||r_i - Qp_i||_2}.$$

For the full derivation and proof of convergence of these update rules, see [12].

3.2.4 *tanh*NMF Algorithm

We also implemented the $tanh$NMF algorithm, proposed by Shen et al. [10]. This method is motivated by the desirable properties of L_0 loss in its robustness to noise. L_0 loss is hard to optimize, and so the authors provide a novel $tanh$-based loss function, and prove that it behaves like L_0 loss and may be optimized using half-quadratic minimization methods.

The objective is defined as

$$\min_{P,Q\geq 0} L(P, Q) = \sum_{i,j} l(E_{ij}) = \sum_{i,j} \tanh\left(\alpha E_{ij}^2\right)$$

where $E = R - QP$ is the matrix of reconstruction errors. This objective is non-convex, so using the framework of half-quadratic minimization, they derive an equivalent, augmented objective

$$J(U, P, Q) = \min_{U,P,Q\geq 0} \sum_{i,j}[U_{ij}E_{ij}^2 + \psi(U_{ij})]$$

where U is the auxiliary variable and $\psi(U_{ij})$ is the conjugate function of $l(E_{ij})$.

When U is fixed, the optimization is equivalent to solving a weighted NMF, where P and Q may be updated by:

$$Q'_{ab} = Q_{ab} \times \frac{(U \odot (RP^T))_{ab}}{(U \odot (QPP^T))_{ab}}$$
$$P'_{ab} = P_{ab} \times \frac{(Q^T(U \odot R))_{ab}}{(Q^T(U \odot (QP)))_{ab}}$$

where $\odot$ denotes the elementwise product of two matrices

Using the half-quadratic minimization algorithm, when P and Q are fixed, the values of U_{ij} can be calculated as:

$$U_{ij} = \alpha[1 - \tanh^2(\alpha|E_{ij}|)] \tag{14}$$

Here, α is a positive constant with value set to

$$\alpha = (\frac{1}{nm}\sum_{ij} E_{ij}^2)^{-1} \tag{15}$$

The $tanh$NMF algorithm alternates updates of Q, P, U and α according to the above rules until convergence.

3.3 Adding Noise to Data

We implemented two different methods to simulate noisy data. Both noise algorithms take an image as input, and a desired proportion of pixels to be affected by the noise. The affect of both noise algorithms are demonstrated in figure 2.

3.3.1 Random White Pixels

The first noise algorithm assigns a random number from 0 to 1 to each pixel. If that number is larger than the specified proportion, then the pixel will be set to a value of 255, making it white. This results in images where a specified proportion p of pixels are white, resulting in images with white dots randomly scattered across them.

3.3.2 Laplacian Noise

We chose to model the effect of Laplacian noise as it violates the assumption of Guassian noise held by classical L_2-norm NMF [12]. The probability density function of the Laplace distribution is

$$f(x; \mu, \lambda) = \frac{1}{2\lambda} exp(-\frac{|x - \mu|}{\lambda})$$

where μ is the location parameter, and λ is the scale parameter. We held the location parameter fixed at 0 for all experiments, but varied the scale parameter. With original data $\hat{R} \in \mathbb{R}^{n \times d}$, we created noisy data R by sampling nd points from the Laplace distribution to yield a noise matrix $N \in \mathbb{R}^{n \times d}$. The noisy matrix R is then generated by the equation $R = \hat{R} + N$. Thus, with a larger values of λ, the noise applied to the image increases, as the spread of points increases away from 0.

3.3.3 Random White Boxes

To examine how our algorithms would respond to contiguous noise, we created an algorithm to generate random white boxes in each image. For each image, the box is placed at a random location in the image. Furthermore, the width of each white box is randomly chosen, and the height is calculated such that the box covers a specified proportion of each image. This means that we can compare directly the effects of adding random white pixels at a specified portion (see 3.3.1), with adding the same amount of noise, except in a contiguous fashion. Importantly, the range of the randomly chosen width is bounded to ensure that the required calculated height does not exceed the image dimensions.

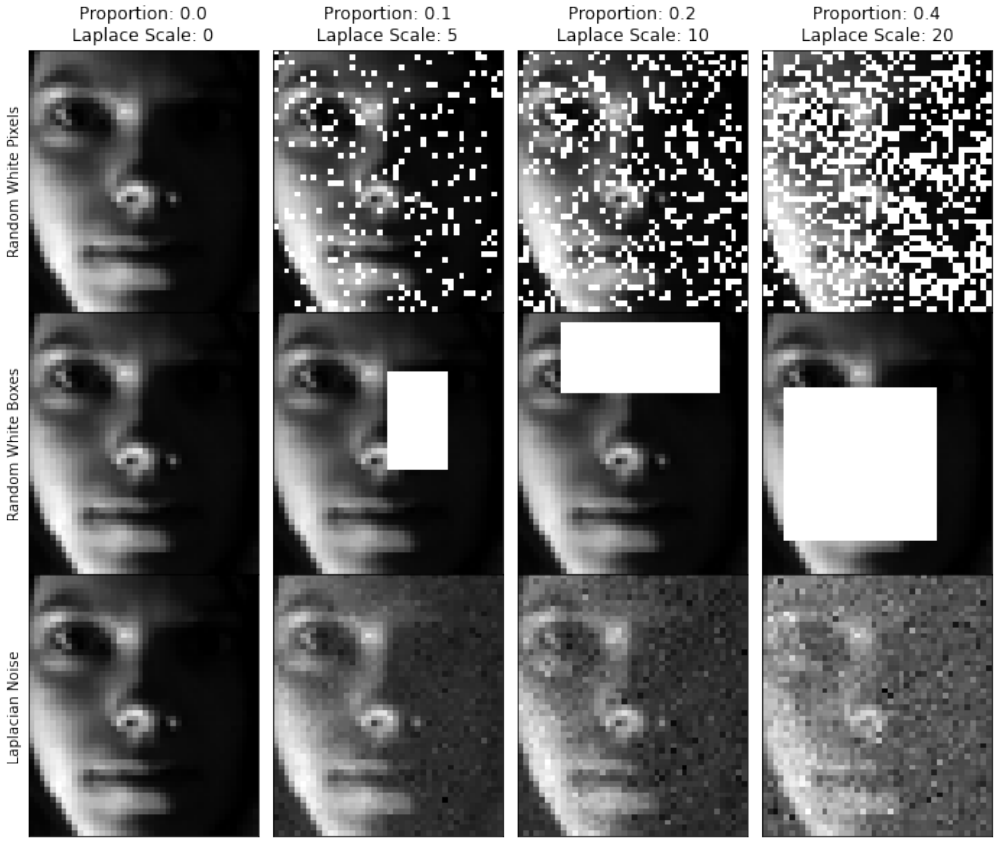

Figure 2: Images demonstrating the effects of our noise algorithms with various levels of noise proportions for white and box noise, and various scales of laplacian noise

3.4 Evaluation Procedure and Metrics

We employed three evaluation metrics to measure the performance of NMF algorithms in response to noise. These were RRE, Accuracy, and Normalised Mutual Information (NMI).

3.4.1 Relative Reconstruction Error

We use relative reconstruction error (RRE) as defined in the assignment guide. We have data matrix R with dictionary Q and coefficients P. Thus RRE is defined as

$$RRE = \frac{\|\hat{R} - QP\|_F}{\|\hat{R}\|_F},$$

where $\hat{R}$ is the data set before noise is added (clean), and Q, P are the factorisation results on R (the data set after adding noise).

3.4.2 Accuracy

To measure the accuracy of our NMF algorithms, we will be using K-Means clustering of the entries in our coefficients matrix P, and measure the proportion of correctly labelled images. Each image is assigned to a cluster label, then the accuracy of the predictions Y_{pred} produced by K-Means on the factorisation result P, can be calculated with respect to the actual labels Y using

$$Acc(Y, Y_{pred}) = \frac{1}{n} \sum_{i=1}^{n} 1\{Y_{pred}(i) == Y(i)\}.$$

3.4.3 Normalised Mutual Information (NMI)

We will also be measuring the Normalised Mutual Information (NMI) between the cluster labels and actual image using the formula

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

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

3.4.4 Evaluation Procedure

For each algorithm described in section 3.2, we employed five-fold cross validation where each fold was a 90% random sample of the original data. All algorithms were measured on the same five folds to ensure differences in performance were not the result of the random sampling procedure. We repeated the cross validation procedure for varying levels of noise introduced by the algorithms described in section 3.3.

4 Experiments

4.1 Hyperparameter Optimisation

Prior to evaluating the NMF algorithms described in section 3.2, we first aimed to determine the optimal number of latent variables k for producing the factorisation $R = QP$ where $R \in \mathbb{R}^{n \times d}$ in terms of two matrices $Q \in \mathbb{R}^{n \times k}$ and $P \in \mathbb{R}^{k \times d}$. It is known that the value of k in such a factorisation can influence the accuracy of the reconstruction, with larger values of k improving accuracy. However, to achieve a compression with the factorisation, it is necessary that k is less than both n and d. Thus we performed 5-fold cross-validation on the ORL data with folds re-sampled at 90% to determine an optimal value for k. We used the L_2-NMF algorithm with RRE as the metric examined. The results are displayed in 3. We found that using $k = 100$ latent variables yielded an acceptable RRE without noise of 6.67%. The run time for $k = 100$ latent variables was also sufficiently fast, and represents a compression on the original data (original data sizes used in this experiment were 360×644 for ORL and 2172×2016 for Yale).

Next, to determine the optimal β value for L_2-norm regularised NMF, we ran 5-fold cross-validation with a constant noise value of $p = 0.2$ for random white pixels added to the ORL data. We found that a β value of 1000 yielded the best results 3, with an RRE of 27.47%. The optimal β value was higher than we expected, although given pixel values range from 0-255 and all noise values for this noise algorithm are added at the upper range of pixel intensity. We subsequently used a β value of 1000 when comparing the different NMF algorithms.

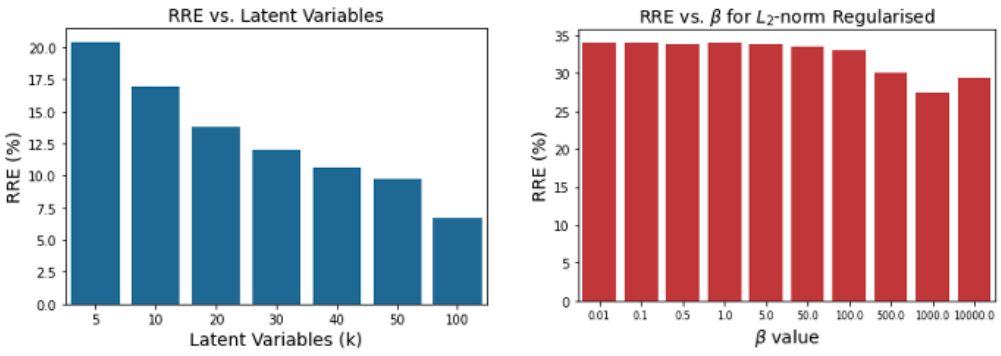

Figure 3: Mean relative reconstruction error (%) from 5-fold cross validation for different numbers of latent variables using the L_2-norm with (left), and for different β values for the L_2 regularised NMF (right).

4.2 Analysis Based on Evaluation Metrics

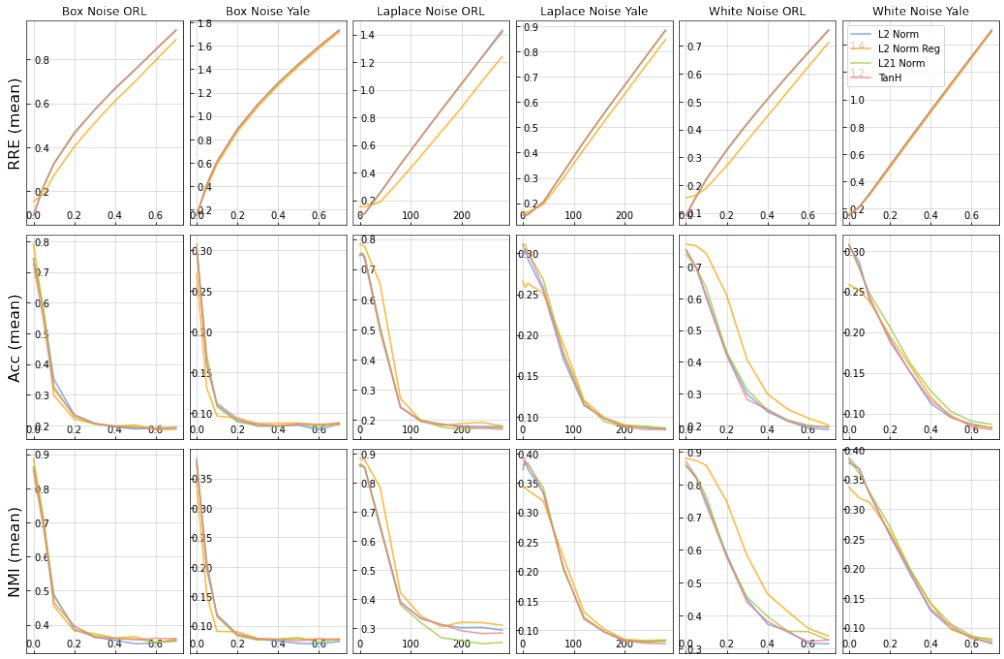

Figure 4: The mean result obtained using both data sets, different noise types, and different NMF algorithms. Each row of subplots represents a different measure of our NMF's performance. Table of results with standard deviations can be found in appendix B

For each noise type, data-set and NMF algorithm, we ran 5 trials of NMF using a randomized 90% portion of a given data set. The mean of our evaluation metrics of each of the 5 trials was calculated, and results are plotted in figure 4, and tables with mean and standard deviations of our results are provided in the appendix of this report.

We can see that the relative reconstruction error (RRE) clearly increases as the noise amount increases, which we'd expect since the addition of noise makes it more difficult for the NMF algorithms to learn the true underlying distribution of the data. All our NMF algorithms seem to perform about the same using this measure, except for the L_2-norm NMF with regulariser, which seems to be a bit more robust to noise compared to the others, particularly with the ORL data. The reduced variance on the Yale data set may be explained by the fact there are more images per person in the Yale dataset compared to the ORL data set[13][14].

Notably, on data contaminated by Laplacian noise, the regularised L_2-NMF still appeared to outperform the $L_{2,1}$-NMF, which was originally designed under the assumption of the data being contaminated by Laplacian noise [12]. Unfortunately, we could not replicate the exact implementation of the $L_{2,1}$-NMF as described in [12], as key details such as the number of latent variables, and the number of iterations were not included. This may have affected $L_{2,1}$-NMF performance.

Using the measure of accuracy in Figure 4, we again see that most of our algorithms perform about the same, but the regularizing term occasionally improves results. The regularizing terms seem to perform particularly well on the Laplace noise ORL, and the white noise ORL trials. Looking at the measure of NMI in Figure 4, we again see that the regularizing term improved results for the ORL data set with white and Laplace noise.

As expected, the addition of a regularizing term decreases over-fitting of the data, making it more robust. In future studies, we could try adding regularizing terms to $L_{2,1}$-norm, $tanh$NMF, or any other NMF algorithm and see how this affects the results. It is worth noting that since the β value for the L_2-regularised NMF was decided using the white pixel noise algorithm, it may have been optimised for that type of noise. Since the white noise algorithm adds noise at the upper range of pixel intensity (pixels are randomly converted to have a value of 255), this may have resulted in a

larger value for β being chosen. In future, we could evaluate the optimal β value across all noise types.

4.3 Analysis Based on Image Reconstructions

In Figure 5, we plot the reconstruction of our faces as images using each NMF algorithm. The first column in the image matrix displays the original image, and the remaining columns show the reconstructed images after training the NMF with various white and box noise amounts. Although it is clear that the addition of noise reduces the quality of our reconstructions, we can not visually distinguish the performance of each algorithm with varying noise.

Figure 5: Comparing each algorithms reconstruction using various noise amounts and both noise types.

Interestingly, with the addition of box noise to our images, our NMF images does sometimes seem to reconstruct images with face like features instead of boxes.

5 Conclusion

We have discussed and successfully built four NMF algorithms, and tested results on two different data sets with various amounts of white, box and laplacian noise. The performance of each algorithms was measured using the metrics of RRE, Accuracy and NMI, and generally speaking, the L_2-norm NMF with regularizer performed best, especially on the ORL data set with white or Laplacian noise.

Since regularization significantly improved results, it would be worthwhile investigating the affect that a regularizing term would have on our other algorithms such as $L_{2,1}$-norm and $tanh$NMF. Additionally, we only ran each NMF trial for 500 steps in our main experiment, and we did not check if our NMF algorithms actually converged. Perhaps we could improve our results by increasing the number of steps, our running our algorithms until some convergence criteria is met. Doing so however would greatly increase the computational demand of our experiment, and we'd have to run it for a significant longer amount of time.

There are also plenty of other NMF algorithms that we discussed in our previous work section that we did not implement, which could also greatly improve results. In particular, the Truncated Cauchy NMF [9] seems very promising in terms of robustness, and we could explore if truncating various other loss functions improve their performance too.

References

[1] S.Velliangiri S.Alagumuthukrishnan, and S Iwin Thankumar joseph. **A Review of Dimensionality Reduction Techniques for Efficient Computation**. Procedia Computer Science. 2019. DOI: 10.1016/j.procs.2020.01.079

[2] Edmund Y. Lam **Non-negative Matrix Factorization for Images with Laplacian Noise** Imaging Systems Laboratory, Department of Electrical and Electronic Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong. DOI:10.1109/APCCAS.2008.4746143

[3] Lijun ZHANG, Zhengguang CHEN, Miao ZHENG, and Xiaofei HE. **Robust non-negative matrix factorization** Front. Electr. Electron. Eng. China. 2011. DOI: 10.1007/s11460-011-0128-0

[4] C. Samson, L. Blanc-Féraud, G. Aubert, and J. Zerubia. **A variational model for image classification and restoration** IEEE Trans. Pattern Anal. Mach. Intell. 2000. DOI: 10.1109/34.857003

[5] A. Ben Hamza and David J. Brady. **Reconstruction of Reflectance Spectra Using Robust Nonnegative Matrix Factorization** IEEE TRANSACTIONS ON SIGNAL PROCESSING. 2006. DOI: 10.1109/TSP.2006.879282

[6] Jorge Nocedal and Stephen J. Wright. **Numerical Optimization** *Second Edition*. Springer. 2006.

[7] Weifeng Liu, Puskal P. Pokharel, and Jose C. Principe. **Correntropy: Properties and Applications in Non-Gaussian Signal Processing**. IEEE TRANSACTIONS ON SIGNAL PROCESSING. 2007. DOI: 10.1109/TSP.2007.896065

[8] Liang Du, Xuan Li, and Yi-Dong Shen. **Robust Nonnegative Matrix Factorization Via Half-Quadratic Minimization** IEEE 12th International Conference on Data Mining. 2012. DOI: 10.1109/ICDM.2012.39

[9] Naiyang Guan, Tongliang Liu, Yangmuzi Zhang, Dacheng Tao, and Larry S. Davis. **Truncated Cauchy Non-negative Matrix Factorization**. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2019. DOI: 10.1109/TPAMI.2017.2777841.

[10] Xingyu Shen, Xiang Zhang, Long Lan, Qing Liao, and Zhigang Luo. **Another Robust NMF: Rethinking the Hyperbolic Tangent Function and Locality Constraint**. IEEE Access. 2019. DOI: 10.1109/ACCESS.2019.2903309.

[11] Daniel Lee, and H. Sebastian Seung. **Algorithms for Non-negative Matrix Factorization**. NeurIPS Proceedings. 2000. DOI:10.1145/2063576.2063676.

[12] Deguang Kong, Chris Ding, and Heng Huang. **Robust nonnegative matrix factorization using L21-norm**. CIKM Proceedings. 2011.

[13] Athinodoros S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. **From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose** IEEE Transactions on Pattern Analysis and Machine Intelligence. 2001. DOI: 10.1109/34.927464

[14] F.S. Samaria and A.C. Harter. **Parameterisation of a stochastic model for human face identification**. Proceedings of 1994 IEEE Workshop on Applications of Computer Vision. 1994. DOI: 10.1109/ACV.1994.341300

A Appendix - Running Our Code

Our code has been split into to main notebooks. Although there is some overlap in the code that is contained within them, one of them contains the code for running the main experiment, and the other contains code for generating figures of this report. Both notebooks should run fine using a standard Jupyter Lab environment. All required libraries to run our code is specified in the first few lines of code of each notebook, where we import the required libraries.

A.1 Notebook for Figures

The jupyter notebook named notebook2 contains code primarily for generating figures for this report. To run the final cell in this notebook, the following CSV files listed bellow are required to be in the same directory as the notebook. These CSV files can be generated using the other notebook (notebook1):

- boxNoise_ORL_results.csv
- boxNoise_YALE_results.csv
- lapNoise_ORL_results.csv
- lapNoise_YALE_results.csv
- whiteNoise_ORL_results.csv
- whiteNoise_YALE_results.csv

B Appendix - Tables of Results

Here we present the detailed results for all evaluation experiments. The means and standard deviations of all experiments are presented. For each experiment, where the metric is RRE, the lowest value is in bold, where the metric is accuracy or NMI, the highest value is in bold. The type of noise and dataset used are mentioned in the caption for each table.

Table 1: Mean and standard deviation of RRE for white noise levels specified by p. Results for ORL data set.

p	L_2-norm	L_2-norm-regularised	$L_{2,1}$-norm	tanh
0	**9.05(0.05)**	15.37(0.1)	9.23(0.03)	9.08(0.05)
0.05	15.86(0.18)	16.34(0.14)	15.77(0.23)	**15.67(0.16)**
0.1	22.22(0.19)	**18.93(0.23)**	22.02(0.24)	22.05(0.29)
0.2	32.68(0.41)	**26.89(0.31)**	32.41(0.41)	32.58(0.38)
0.3	42.08(0.52)	**35.9(0.52)**	41.86(0.49)	42.06(0.57)
0.4	50.86(0.6)	**44.74(0.56)**	50.8(0.73)	50.9(0.57)
0.5	59.37(0.72)	**53.59(0.77)**	59.33(0.72)	59.46(0.78)
0.6	67.67(0.88)	**62.5(0.84)**	67.54(0.87)	67.6(0.79)
0.7	75.7(0.95)	**71.21(0.95)**	75.63(0.94)	75.67(0.9)

Table 2: Mean and standard deviation of Accuracy (%) for white noise levels specified by p. Results for ORL data set.

p	L_2-norm	L_2-norm-regularised	$L_{2,1}$-norm	tanh
0	75.28(2.37)	**77.06(2.16)**	74.0(0.92)	75.44(2.23)
0.05	69.5(2.67)	**76.56(1.15)**	69.78(1.49)	69.83(1.65)
0.1	60.89(1.53)	**74.17(1.45)**	63.22(2.41)	60.06(1.35)
0.2	42.5(1.95)	**61.0(1.9)**	43.17(2.33)	42.06(2.98)
0.3	29.83(1.46)	**40.72(0.78)**	31.28(1.21)	28.17(0.91)
0.4	24.33(0.85)	**29.89(1.72)**	24.83(0.54)	25.06(1.26)
0.5	21.33(1.34)	**25.06(0.57)**	21.72(1.88)	21.44(0.73)
0.6	19.22(0.48)	**22.28(1.23)**	20.06(0.67)	19.72(0.61)
0.7	18.67(0.92)	**20.06(1.3)**	19.39(0.41)	19.5(0.89)

C Appendix - Contributions

All three members contributed equally to this assignment. We split the work by each implementing different NMF algorithms, and by each working on separate parts of the project, such as the noise algorithms and evaluation algorithms. All members contributed to writing the final report.

Table 3: Mean and standard deviation of NMI (%) for white noise levels specified by p. Results for ORL data set.

p	L_2-norm	L_2-norm-regularised	$L_{2,1}$-norm	tanh
0	85.84(1.1)	**87.98(1.06)**	85.71(0.67)	86.96(1.05)
0.05	81.9(1.71)	**87.32(0.82)**	82.4(0.9)	82.33(1.26)
0.1	74.82(1.39)	**85.83(0.74)**	76.33(1.62)	73.54(0.81)
0.2	58.63(0.95)	**74.79(1.46)**	57.74(3.14)	57.99(2.47)
0.3	44.94(2.8)	**58.3(0.44)**	45.61(1.86)	43.96(1.77)
0.4	37.36(2.37)	**46.63(2.81)**	39.76(1.06)	38.22(1.45)
0.5	35.25(2.04)	**41.17(1.24)**	35.19(1.96)	34.89(2.09)
0.6	31.51(2.32)	**36.07(1.49)**	35.05(1.81)	32.02(1.94)
0.7	31.35(1.13)	**33.69(2.71)**	32.44(2.34)	32.56(1.34)

Table 4: Mean and standard deviation of RRE (%) for white noise levels specified by p. Results for Yale data set.

p	L_2-norm	L_2-norm-regularised	$L_{2,1}$-norm	tanh
0	**15.07(0.09)**	16.27(0.09)	15.43(0.1)	15.15(0.14)
0.05	21.31(0.16)	**21.08(0.14)**	21.39(0.15)	21.21(0.17)
0.1	30.96(0.19)	**29.89(0.23)**	30.72(0.24)	30.92(0.24)
0.2	51.55(0.38)	**49.98(0.36)**	51.07(0.39)	51.5(0.41)
0.3	71.83(0.56)	**70.27(0.54)**	71.42(0.63)	71.8(0.59)
0.4	91.79(0.76)	**90.3(0.74)**	91.49(0.7)	91.76(0.77)
0.5	111.58(0.92)	**110.15(0.89)**	111.31(0.96)	111.58(0.9)
0.6	131.24(1.07)	**129.99(1.03)**	131.0(1.06)	131.21(1.11)
0.7	150.75(1.22)	**149.67(1.23)**	150.62(1.2)	150.8(1.22)

Table 5: Mean and standard deviation of Accuracy (%) for white noise levels specified by p. Results for Yale data set.

p	L_2-norm	L_2-norm-regularised	$L_{2,1}$-norm	tanh
0	30.61(0.72)	25.84(1.03)	**30.81(1.29)**	30.75(1.23)
0.05	**28.6(1.18)**	25.15(0.55)	27.77(0.46)	28.04(1.23)
0.1	24.01(1.03)	23.8(0.39)	**24.81(1.42)**	24.43(1.05)
0.2	19.19(1.07)	19.54(0.43)	**20.64(0.86)**	18.82(1.08)
0.3	15.06(0.83)	15.87(0.48)	**16.18(0.78)**	15.09(0.44)
0.4	11.25(0.46)	12.03(0.89)	**12.79(0.59)**	11.63(0.58)
0.5	9.48(0.26)	9.56(0.35)	**10.25(0.59)**	9.75(0.36)
0.6	8.72(0.21)	8.54(0.43)	**9.07(0.1)**	8.38(0.23)
0.7	8.22(0.27)	8.29(0.24)	**8.66(0.24)**	8.01(0.2)

Table 6: Mean and standard deviation of NMI (%) for white noise levels specified by p. Results for Yale data set.

p	L_2-norm	L_2-norm-regularised	$L_{2,1}$-norm	tanh
0	37.75(1.46)	33.64(2.05)	38.21(1.95)	**38.54(1.28)**
0.05	**36.83(1.53)**	31.86(1.09)	36.06(1.85)	36.7(1.34)
0.1	32.59(1.41)	31.16(1.03)	**32.89(1.53)**	32.6(1.87)
0.2	25.58(1.16)	26.16(0.41)	**27.06(1.27)**	25.7(1.15)
0.3	19.04(0.72)	19.8(0.98)	**19.79(0.97)**	19.46(0.31)
0.4	12.86(0.73)	14.11(0.93)	**14.13(0.59)**	13.15(0.63)
0.5	9.82(0.34)	9.69(0.63)	**10.58(0.74)**	10.31(0.75)
0.6	8.44(0.26)	8.44(0.23)	**8.63(0.27)**	8.14(0.39)
0.7	7.27(0.33)	**8.09(0.7)**	7.77(0.52)	7.47(0.52)

Table 7: Mean and standard deviation of RRE (%) for box noise levels specified by p. Results for ORL data set.

p	L_2-norm	L_2-norm-regularised	$L_{2,1}$-norm	tanh
0	**9.05(0.05)**	15.37(0.1)	9.23(0.03)	9.08(0.05)
0.05	22.48(0.37)	**18.74(0.21)**	22.39(0.42)	22.28(0.32)
0.1	32.61(0.4)	**27.21(0.32)**	32.37(0.38)	32.68(0.4)
0.2	46.68(0.48)	**40.31(0.44)**	46.55(0.55)	46.05(0.83)
0.3	56.95(0.58)	**51.07(0.69)**	57.0(0.76)	57.15(0.63)
0.4	66.84(0.75)	**61.01(0.86)**	66.64(0.37)	66.55(0.43)
0.5	75.45(0.84)	**69.98(0.64)**	75.35(0.71)	75.4(0.73)
0.6	84.44(1.12)	**79.56(1.06)**	84.48(0.9)	84.42(0.97)
0.7	93.28(1.01)	**89.04(1.04)**	93.23(1.03)	93.28(1.03)

Table 8: Mean and standard deviation of Accuracy (%) for box noise levels specified by p. Results for ORL data set.

p	L_2-norm	L_2-norm-regularised	$L_{2,1}$-norm	tanh
0	74.39(1.44)	**79.0(2.76)**	74.28(1.03)	72.61(1.72)
0.05	56.78(1.84)	56.56(4.11)	**57.44(4.52)**	53.78(2.37)
0.1	**35.11(3.44)**	29.89(1.34)	32.78(2.43)	31.89(0.92)
0.2	**23.56(0.79)**	22.11(1.39)	22.94(1.49)	23.5(1.72)
0.3	**20.78(1.51)**	20.61(1.09)	20.5(0.71)	20.72(0.33)
0.4	19.56(1.15)	19.89(0.8)	**19.94(1.16)**	19.78(0.83)
0.5	18.94(0.89)	**20.22(0.87)**	19.5(1.16)	19.33(0.45)
0.6	19.17(0.46)	18.72(1.05)	**19.22(0.73)**	19.11(0.98)
0.7	**19.39(0.73)**	18.78(1.0)	19.33(0.76)	19.28(0.52)

Table 9: Mean and standard deviation of NMI (%) for box noise levels specified by p. Results for ORL data set.

p	L_2-norm	L_2-norm-regularised	$L_{2,1}$-norm	tanh
0	86.07(0.89)	**88.78(1.22)**	86.34(1.01)	85.23(1.29)
0.05	69.73(0.83)	**71.03(3.18)**	70.11(3.7)	68.04(2.23)
0.1	**48.89(3.55)**	45.52(1.27)	48.29(1.49)	46.56(1.14)
0.2	38.69(1.15)	38.2(1.43)	39.54(0.91)	**39.67(1.73)**
0.3	36.41(1.86)	**37.4(1.41)**	36.74(1.49)	36.3(1.42)
0.4	35.46(2.36)	**36.15(1.08)**	36.01(0.86)	36.02(1.32)
0.5	34.44(1.79)	**36.47(1.03)**	35.82(1.19)	35.6(1.07)
0.6	34.66(1.53)	34.81(0.93)	34.93(1.78)	**36.06(0.98)**
0.7	35.48(1.42)	35.79(0.95)	35.14(1.11)	**35.94(1.07)**

Table 10: Mean and standard deviation of RRE (%) for box noise levels specified by p. Results for Yale data set.

p	L_2-norm	L_2-norm-regularised	$L_{2,1}$-norm	tanh
0	**15.07(0.09)**	16.27(0.09)	15.43(0.1)	15.15(0.14)
0.05	41.51(0.28)	**38.99(0.3)**	41.4(0.44)	41.45(0.35)
0.1	61.1(0.36)	**58.71(0.67)**	61.02(0.49)	61.13(0.56)
0.2	88.76(0.78)	**86.48(0.58)**	88.62(0.76)	88.67(0.8)
0.3	109.96(1.05)	**107.79(1.07)**	109.86(1.0)	109.85(0.96)
0.4	128.06(1.08)	**126.08(1.03)**	128.06(1.19)	128.15(1.07)
0.5	144.38(1.26)	**142.37(1.22)**	144.23(1.18)	144.25(1.32)
0.6	159.21(1.29)	**157.38(1.25)**	159.18(1.32)	159.26(1.37)
0.7	173.34(1.4)	**171.82(1.49)**	173.32(1.42)	173.38(1.49)

Table 11: Mean and standard deviation of Accuracy (%) for box noise levels specified by p. Results for Yale data set.

p	L_2-norm	L_2-norm-regularised	$L_{2,1}$-norm	tanh
0	30.48(2.39)	27.13(1.84)	**30.76(1.37)**	30.3(1.1)
0.05	15.87(0.54)	13.05(0.78)	**17.08(1.32)**	16.0(1.04)
0.1	10.98(0.71)	9.62(0.22)	10.77(0.3)	**11.2(0.85)**
0.2	9.03(0.44)	**9.35(0.49)**	8.9(0.48)	9.31(0.47)
0.3	8.59(0.3)	**8.72(0.19)**	8.4(0.27)	8.36(0.24)
0.4	8.41(0.35)	**8.74(0.36)**	8.31(0.32)	8.44(0.28)
0.5	8.47(0.2)	**8.75(0.32)**	8.59(0.3)	8.64(0.55)
0.6	7.94(0.43)	8.54(0.22)	8.27(0.26)	**8.58(0.22)**
0.7	8.64(0.25)	8.49(0.24)	**8.82(0.42)**	8.73(0.19)

Table 12: Mean and standard deviation of NMI (%) for box noise levels specified by p. Results for Yale data set.

p	L_2-norm	L_2-norm-regularised	$L_{2,1}$-norm	tanh
0	37.92(2.01)	34.21(1.55)	37.48(1.34)	**38.39(1.28)**
0.05	19.52(0.94)	15.22(0.87)	**20.93(1.52)**	20.17(0.6)
0.1	**11.9(0.57)**	9.06(0.72)	11.54(0.64)	11.77(0.99)
0.2	8.57(0.55)	**8.99(0.81)**	8.28(0.45)	8.56(0.63)
0.3	7.72(0.46)	**7.88(0.48)**	7.59(0.79)	7.82(0.78)
0.4	7.43(0.53)	7.74(0.88)	**7.76(0.48)**	7.67(0.61)
0.5	7.03(0.14)	7.77(0.34)	**8.02(0.22)**	7.45(0.51)
0.6	6.97(0.34)	7.75(0.32)	7.23(0.55)	**7.79(0.33)**
0.7	7.36(0.46)	7.55(0.58)	**7.81(0.25)**	7.76(0.2)

Table 13: Mean and standard deviation of RRE (%) for laplacian noise levels with scale parameter λ. Results for ORL data set.

λ	L_2-norm	L_2-norm-regularised	$L_{2,1}$-norm	tanh
0	**9.05(0.05)**	15.37(0.1)	9.23(0.03)	9.08(0.05)
5	**9.39(0.05)**	15.36(0.1)	9.51(0.05)	9.42(0.05)
10	10.48(0.06)	15.33(0.09)	10.54(0.04)	**10.47(0.05)**
40	25.41(0.11)	**18.67(0.14)**	25.38(0.12)	25.43(0.16)
80	46.02(0.34)	**34.91(0.12)**	46.06(0.23)	45.95(0.25)
120	65.83(0.42)	**52.04(0.46)**	65.9(0.62)	65.7(0.7)
160	85.37(0.4)	**69.67(0.39)**	85.39(0.87)	85.2(0.38)
200	104.56(0.97)	**87.12(0.7)**	104.72(0.8)	104.13(0.77)
240	123.77(1.14)	**106.26(0.46)**	123.43(0.6)	123.57(1.06)
280	143.31(1.04)	**124.15(0.91)**	142.15(0.9)	141.92(0.4)

Table 14: Mean and standard deviation of Accuracy (%) for laplacian noise levels with scale parameter λ. Results for ORL data set.

λ	L_2-norm	L_2-norm-regularised	$L_{2,1}$-norm	tanh
0	74.56(1.87)	**78.28(1.35)**	75.0(2.65)	74.39(1.13)
5	75.06(1.86)	**78.22(2.71)**	75.44(1.06)	75.11(1.44)
10	74.17(2.14)	**77.39(2.24)**	73.17(0.97)	73.39(2.12)
40	50.67(2.64)	**65.11(2.37)**	50.72(2.41)	49.28(2.54)
80	24.11(0.97)	**27.06(1.34)**	24.11(2.21)	24.22(0.77)
120	**19.89(1.5)**	19.78(0.51)	19.61(0.33)	19.33(1.05)
160	18.39(0.79)	18.11(0.64)	17.61(0.85)	**18.67(0.59)**
200	17.89(1.0)	**18.72(0.65)**	16.72(0.98)	17.44(0.27)
240	17.78(0.88)	**19.17(0.5)**	17.39(1.11)	17.22(1.12)
280	17.72(0.64)	**17.89(0.69)**	17.28(0.67)	16.83(0.45)

Table 15: Mean and standard deviation of NMI (%) for laplacian noise levels with scale parameter λ. Results for ORL data set.

λ	L_2-norm	L_2-norm-regularised	$L_{2,1}$-norm	tanh
0	86.2(1.3)	**88.5(0.98)**	86.47(1.13)	85.86(1.07)
5	85.78(0.99)	**88.18(1.24)**	86.35(0.57)	85.84(1.11)
10	85.45(1.65)	**87.91(1.3)**	84.92(1.16)	85.15(1.59)
40	65.5(1.66)	**78.55(0.99)**	65.54(2.29)	64.61(1.97)
80	39.14(1.27)	**42.25(1.66)**	38.09(2.2)	38.74(0.71)
120	33.37(1.46)	**34.29(2.41)**	31.98(1.89)	33.49(2.18)
160	31.24(0.66)	30.57(1.7)	26.79(2.94)	**31.38(1.47)**
200	30.22(1.5)	**32.08(0.43)**	25.79(0.9)	29.16(1.01)
240	30.35(1.01)	**31.97(1.97)**	24.69(2.12)	28.19(1.95)
280	29.36(1.21)	**31.08(1.26)**	25.14(1.84)	28.38(0.93)

Table 16: Mean and standard deviation of RRE (%) for laplacian noise levels with scale parameter λ. Results for Yale data set.

λ	L_2-norm	L_2-norm-regularised	$L_{2,1}$-norm	tanh
0	**15.07(0.09)**	16.27(0.09)	15.43(0.1)	15.15(0.14)
5	**15.14(0.09)**	16.31(0.1)	15.46(0.1)	15.21(0.14)
10	**15.39(0.08)**	16.41(0.1)	15.61(0.12)	15.43(0.13)
40	20.34(0.11)	**19.45(0.08)**	20.45(0.14)	20.33(0.13)
80	32.31(0.13)	**29.89(0.11)**	32.35(0.13)	32.34(0.17)
120	44.39(0.18)	**41.35(0.19)**	44.41(0.15)	44.38(0.16)
160	55.72(0.33)	**52.39(0.25)**	55.77(0.23)	55.73(0.26)
200	66.71(0.25)	**63.21(0.27)**	66.74(0.32)	66.73(0.27)
240	77.68(0.34)	**74.1(0.31)**	77.69(0.33)	77.62(0.33)
280	88.46(0.36)	**84.81(0.38)**	88.36(0.31)	88.61(0.37)

Table 17: Mean and standard deviation of Accuracy (%) for laplacian noise levels with scale parameter λ. Results for Yale data set.

λ	L_2-norm	L_2-norm-regularised	$L_{2,1}$-norm	tanh
0	29.78(0.81)	26.56(0.32)	**31.12(2.08)**	31.02(1.59)
5	30.53(1.26)	25.84(0.87)	**31.09(1.84)**	29.89(1.31)
10	29.29(0.75)	26.32(0.87)	30.12(2.0)	**30.16(1.12)**
40	25.44(1.16)	25.11(0.66)	**26.83(1.19)**	25.88(1.39)
80	16.97(0.63)	**18.84(0.84)**	17.82(1.25)	17.53(0.32)
120	11.37(0.43)	**12.03(0.55)**	11.81(0.42)	11.46(0.71)
160	9.79(0.28)	**9.84(0.4)**	9.37(0.34)	9.71(0.47)
200	8.87(0.51)	**8.98(0.12)**	8.71(0.22)	8.76(0.24)
240	8.63(0.2)	**8.79(0.27)**	8.78(0.12)	8.4(0.22)
280	8.4(0.29)	8.55(0.22)	**8.6(0.43)**	8.45(0.22)

Table 18: Mean and standard deviation of NMI (%) for laplacian noise levels with scale parameter λ. Results for Yale data set.

λ	L_2-norm	L_2-norm-regularised	$L_{2,1}$-norm	tanh
0	37.3(1.36)	34.33(1.16)	38.17(1.27)	**39.18(1.72)**
5	**38.81(1.69)**	34.1(1.16)	38.01(1.46)	38.26(1.7)
10	37.34(0.69)	33.75(1.68)	38.01(1.92)	**38.4(1.04)**
40	33.39(2.4)	31.97(0.9)	**34.28(1.66)**	33.4(1.78)
80	20.38(0.6)	**22.33(1.84)**	20.16(1.36)	20.64(1.27)
120	12.0(0.85)	**13.09(0.85)**	12.23(0.69)	11.89(1.04)
160	9.66(0.67)	**10.16(0.64)**	9.7(0.63)	9.7(0.54)
200	8.33(0.6)	**8.41(0.78)**	8.08(0.68)	8.18(0.6)
240	8.07(0.83)	**8.23(0.46)**	7.93(0.28)	7.79(0.37)
280	8.16(0.11)	**8.34(0.46)**	8.18(0.46)	7.68(0.69)