Analysis of the NMF algorithms in Facial Recognition

Members: TRINH Van Hoan
VUONG Tung Duong

Outline

- Introduction
- What is Non-Negative Matrix Factorization (NMF)?
  - NMF with the L2-norm
  - Algorithms
  - NMF with the KL-divergence
  - Modification of the algorithm
  - Theoretical results
- Experimental Setup
- Experimental Methodology and Results
  - Classification with SVM
  - Influence of Number of Components
  - Influence of Noise
- Conclusion

What is Non-Negative Matrix Factorization (NMF)?

- Given non-negative $n \times m$ matrix V, i.e. $V_{ij} \geq 0 \forall i, j$.
- An "approximate" factorization of this matrix $V \approx WH$ is called NMF if $W \in R^{n \times k}$ and $H \in R^{k \times m}$ are non-negative matrices.
- Intuition: W contains a basis for linear approximation of data in V in k −dimension.

(Lee, Seung. 2000)

NMF with the L2-norm

- The objective function is the Euclidean distance:

$\| X - WH \|^2$, w.r.t W, H with constraint $W, H \geq 0$.

Algorithms

- Optimize alternatively over one of the two factors, W or H, while keeping the other fixed.
- Additive update rule based on gradient descent:
  - $W \leftarrow W - \eta_W \nabla_W F$
  - $H \leftarrow H - \eta_H \nabla_H F$

Algorithms(cont'd)

- After derivation:
  - $\nabla_W F = -2VH^T + 2WHH^T$
  - $\nabla_H F = -2W^T V + 2W^T WH$
- Chosing the appropriate step size:
  - $(\eta_W)_{uj} = \dfrac{W_{uj}}{(WHH^T)_{uj}}$
  - $(\eta_H)_{uj} = \dfrac{H_{uj}}{(W^T WH)_{uj}}$

Algorithms(cont'd)

- The algorithm becomes multiplicative:
  - $W_{iu} \leftarrow W_{iu} \frac{(VH^T)_{iu}}{(WHH^T)_{iu}}$
  - $H_{uj} \leftarrow H_{uj} \frac{(W^T V)_{uj}}{(W^T WH)_{uj}}$
- Ensure non-negativity.
- Ensure the non-increasing property of the loss function.

NMF with the KL-divergence

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

constrained to $W \geq 0$ and $H \geq 0$.

- With the same workflow:
  - $H_{aj} \leftarrow \dfrac{\sum_i v_i \frac{W_{ia}H_{aj}}{W_i H_j}}{\sum_i W_{ia}}$
  - Similarly to W.

Modification of the algorithm

- If $\sum_i W_{ia} = 0$, skip the update for H_{aj}.
- If $W_i H_j = 0$ for some i:
  - If $v_i > 0$, update H_{aj} to a constant positive number.
  - If $v_i = 0$, replace the part $v_i \frac{W_{ia} H_{aj}}{W_i H_j}$ by 0.
- Ensure the non-negativity.
- Ensure the non-increasing of the loss function.

Modification of the algorithm(cont'd)

- Do the normalization on the W's columns.
  - $H_{aj}^{t+1,u} = \dfrac{\sum_i v_i \frac{W_{ia} H_{aj}^t}{W_i H_j^t}}{\sum_i W_{ia}}$
  - After get $W^{t+1,u}, H^{t+1,u}$, normalize to W^{t+1}, H^{t+1} such that W^{t+1}'s columns' sum is 1 or 0.

Theoretical Results

- Under the modifications, there exists a limit point W^*, H^*.
- For every limit point W^*, H^*, if $H^*_{aj} > 0$ then its partial derivative is 0: $\nabla_{H_{aj}} F(W^*, H^*) = 0$

See report for details

Experimental Setup

1. NMF implementation
2. SVD-based initialization
3. Noise
   - Salt and Pepper Noise
   - Gaussian Noise
   - Laplace Noise
4. Metrics
5. Datasets

NMF implementation

In Python, implement based on (Lee, Seung. 2000)

1. L2NMF update rules

$$H_{a\mu} \leftarrow H_{a\mu} \frac{(W^T V)_{a\mu}}{(W^T W H)_{a\mu}} \qquad W_{ia} \leftarrow W_{ia} \frac{(V H^T)_{ia}}{(W H H^T)_{ia}}$$

2. KLNMF update rules

$$H_{a\mu} \leftarrow H_{a\mu} \frac{\sum_i W_{ia} V_{i\mu}/(WH)_{i\mu}}{\sum_k W_{ka}} \qquad W_{ia} \leftarrow W_{ia} \frac{\sum_\mu H_{a\mu} V_{i\mu}/(WH)_{i\mu}}{\sum_\nu H_{a\nu}}$$

See code for details

SVD-based initialization

Method 1: Standard randomization

- Sample from a Uniform or Gaussian distribution

Method 2: SVD-based

- NNDSVD (C. Boutsidis, E. Gallopoulos. 2007)

NNDSVD initialization of nonnegative matrix, in MATLAB notation

Inputs: Matrix $A \in \mathbb{R}_+^{m\times n}$, integer $k < \min(m,n)$.
Output: Rank-k nonnegative factors $W \in \mathbb{R}_+^{m\times k}$, $H \in \mathbb{R}_+^{k\times n}$.
1. Compute the largest k singular triplets of A: $[U,S,V] = \mathtt{psvd}(A,k)$
2. Initialize $W(:,1) = \mathtt{sqrt}(S(1,1)) * U(:,1)$ and $H(1,:) = \mathtt{sqrt}(S(1,1)) * V(:,1)'$
for $j = 2:k$
3. $x = U(:,j)$; $y = V(:,j)$;
4. $xp = \mathtt{pos}(x)$; $xn = \mathtt{neg}(x)$; $yp = \mathtt{pos}(y)$; $yn = \mathtt{neg}(y)$;
5. $xpnrm = \mathtt{norm}(xp)$; $ypnrm = \mathtt{norm}(yp)$; $mp = xpnrm * ypnrm$;
6. $xnnrm = \mathtt{norm}(xn)$; $ynnrm = \mathtt{norm}(yn)$; $mn = xnnrm * ynnrm$;
7. **if** $mp > mn$, $u = xp/xpnrm$; $v = yp/ypnrm$; $sigma = mp$;
   else $u = xn/xnnrm$; $v = yn/ynnrm$; $sigma = mn$; **end**
8. $W(:,j) = \mathtt{sqrt}(S(j,j) * sigma) * u$ and $H(j,:) = \mathtt{sqrt}(S(j,j) * sigma) * v'$;
end

See code for details

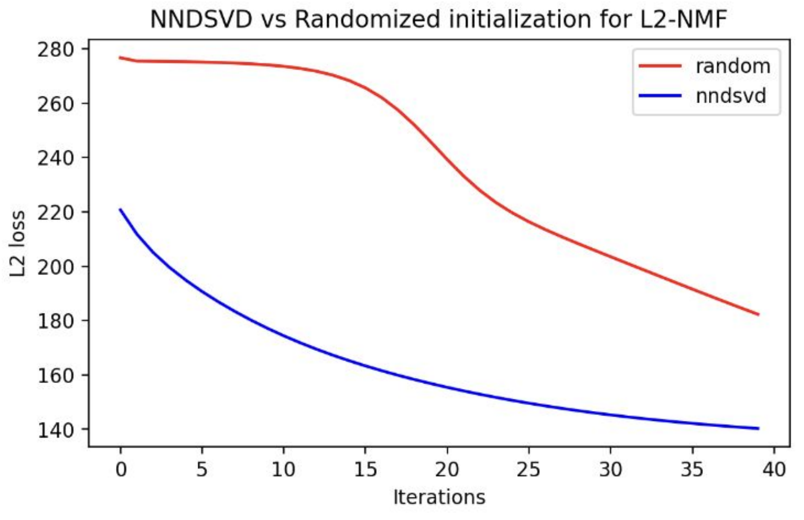
SVD-based initialization

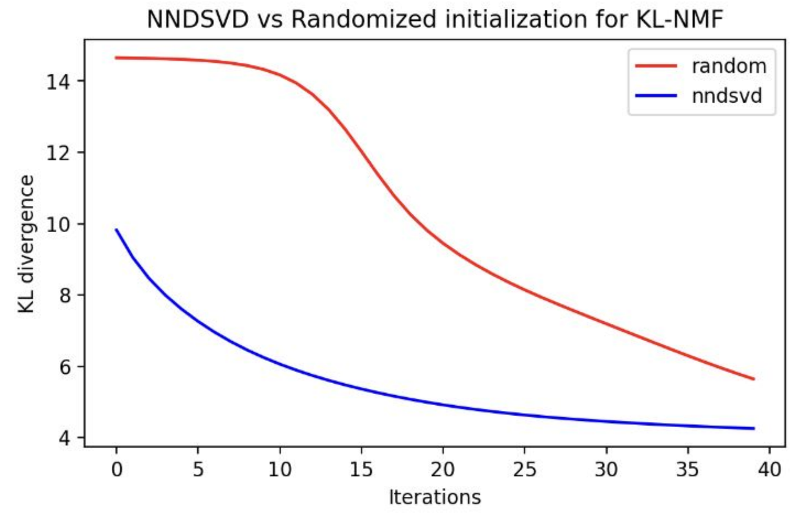
Method 1: Standard randomization

- Sample from a Uniform or Gaussian distribution

Method 2: SVD-based

- NNDSVD (C. Boutsidis, E. Gallopoulos. 2007)





Noise

In practice, the images are corrupted with noise

Three common types:

1. Salt and Pepper Noise:

- generated by malfunctioning of pixel elements in camera sensors or errors in conversion process,...
- commonly corrupt by 255(salt), 0(pepper)
- parameters:
  - p: noise level
  - s_vs_p: salt vs pepper ratio

See code for details

Noise

In practice, the images are corrupted with noise

Three common types:

1. Salt and Pepper Noise:

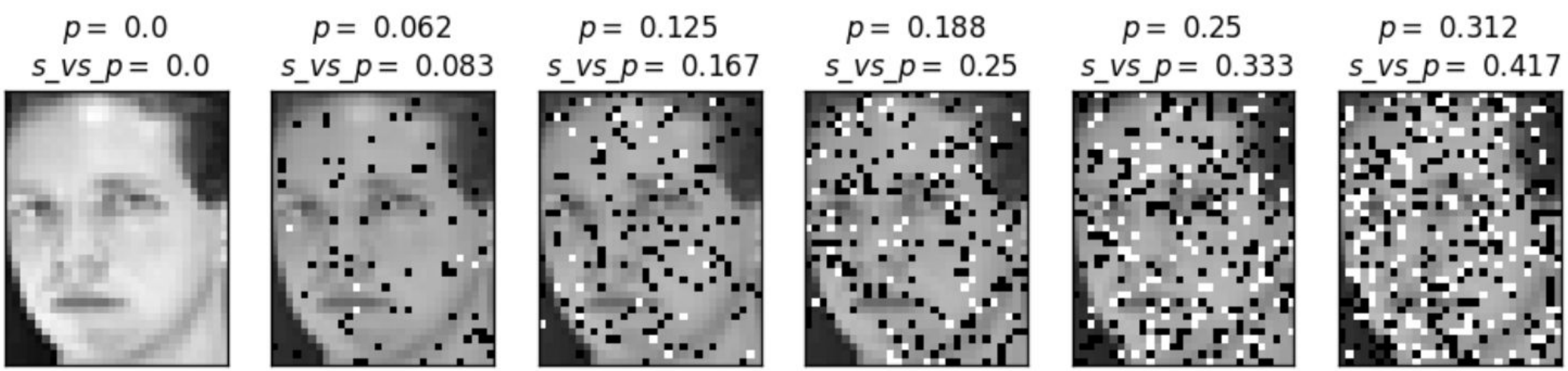


Figure 2: Image with Salt and Pepper noise with different values of hyper-parameters p and s_vs_p

Noise

In practice, the images are corrupted with noise

Three common types:

2. Gaussian noise:

- usually happens in amplifiers or detectors, generates disturbs in the gray values in the digital images,...

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

- parameters: g, sigma, mu

See code for details

Noise

In practice, the images are corrupted with noise

Three common types:

2. Gaussian noise:

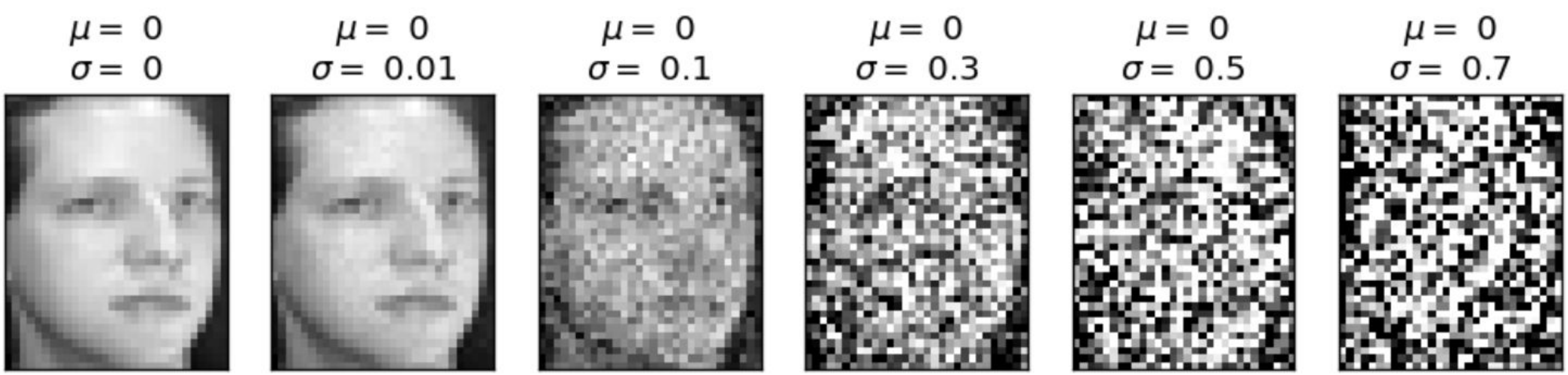


Figure 4: Image with Gaussian noise with different values of σ

Noise

In practice, the images are corrupted with noise

Three common types:

2. Gaussian noise vs Laplacian noise:

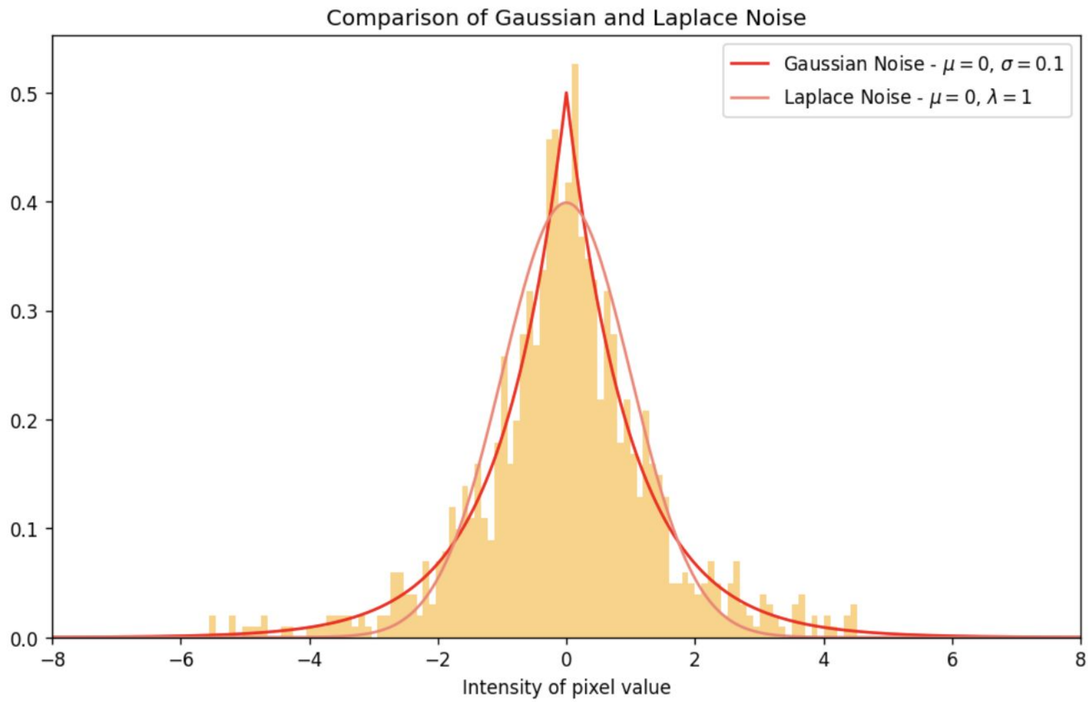


Figure 3: PDF of Gaussian and Laplace noise

Noise

In practice, the images are corrupted with noise

Three common types:

3. Laplacian noise:

- similar to Gaussian noise, but is from the Laplace Distribution

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

- parameter: mu, lambda

See code for details

Noise

In practice, the images are corrupted with noise

Three common types:

3. Laplacian noise:

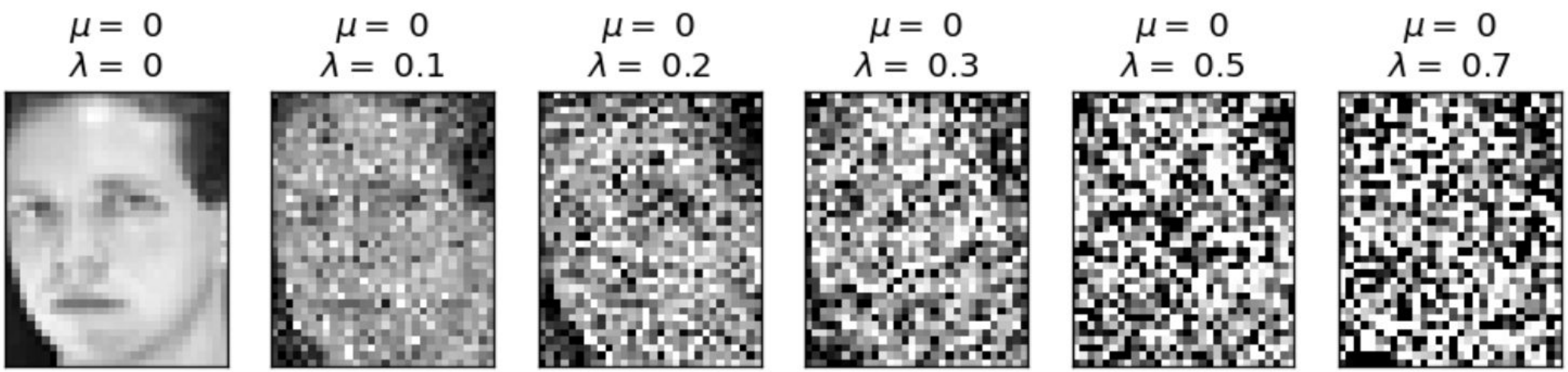


Figure 5: Image with Laplace noise with different values of λ

Metrics

1. Relative Reconstruction Errors (RRE)

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

2. Average Accuracy

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

3. Normalized Mutual Information (NMI)

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

See code for details

Datasets

1. Yale face dataset
   - sample 165 images of 15 subjects
   - under different poses and illumination conditions
   - split into 150 images for training and 15 images for testing
2. ORL face dataset
   - sample 400 images of 40 subjects
   - under different lighting, facial expressions and facial details
   - split into 360 images for training and 40 images for testing

Experimental Methodology and Results

1. Classification with SVM
2. Influence of Number of Components
3. Influence of Noise

Classification with SVM

Three different scenarios

1. Vanilla SVM:
   - directly train SVM and evaluate
2. L2NMF+SVM
   - fit L2NMF using the training images
   - use it to reduce the dimensions of train and test set
   - train SVM and evaluate
3. KLNMF+SVM
   - similar to L2NMF+SVM, but using NMF with the KL divergence

Classification with SVM

Three different scenarios

1. Vanilla SVM
2. L2NMF+SVM
3. KLNMF+SVM

	ORL					
Model	Fit+Train time	Predict time	Accuracy	Precision	Recall	F1-score
Vanila SVM	0+12.194s	0.020s	**0.95**	**0.93**	**0.95**	**0.93**
L2NMF+SVM	2.165+3.367s	**0.003s**	0.93	0.89	0.93	0.90
KLNMF+SVM	**0.345+3.359s**	0.004s	0.93	0.89	0.93	0.90

	Yale					
Model	Fit+Train time	Predict time	Accuracy	Precision	Recall	F1-score
Vanila SVM	0+15.959s	0.016s	**0.73**	**0.66**	**0.73**	**0.68**
L2NMF+SVM	**0.803+0.695s**	**0.001s**	0.67	0.56	0.67	0.59
KLNMF+SVM	1.176+0.794s	**0.001s**	0.67	0.56	0.67	0.59

Table 1: Table results for experiment with SVM

Influence of Number of Components

- Influence of the reduced rank k
- Add Gaussian noise with sigma=0.05 to images data
- No. components range from 10 to 80

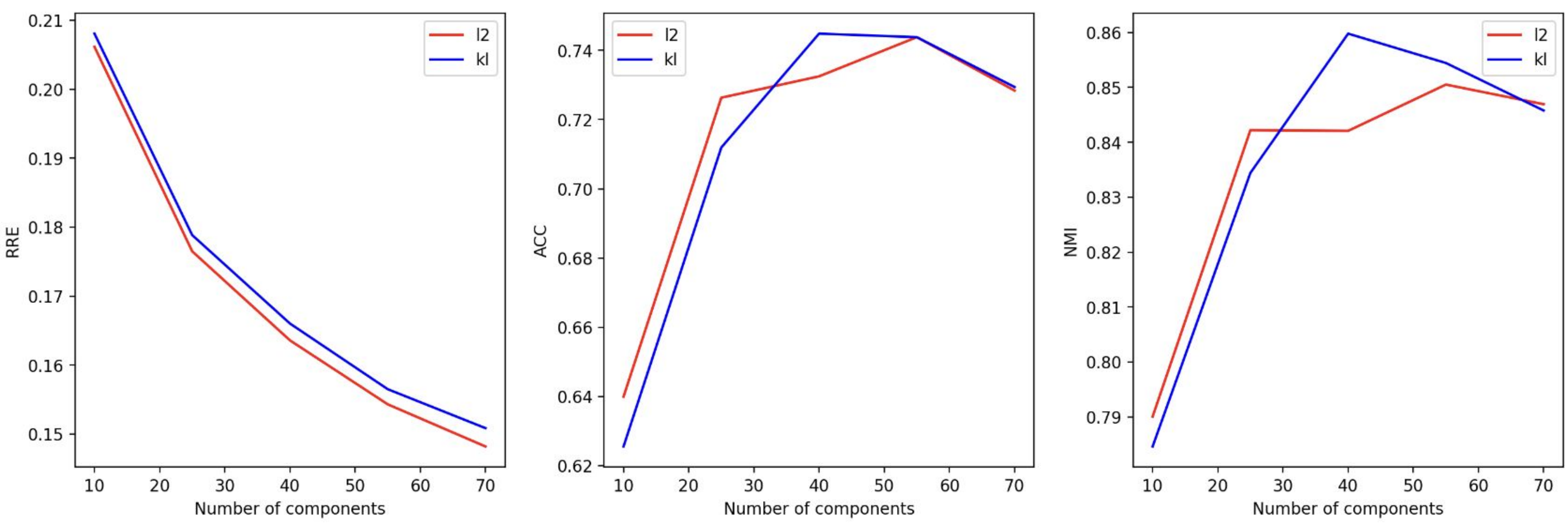


Figure 6: Influence of number of components on RRE, ACC and NMI

Influence of Number of Components

Each experiment was executed 3 times

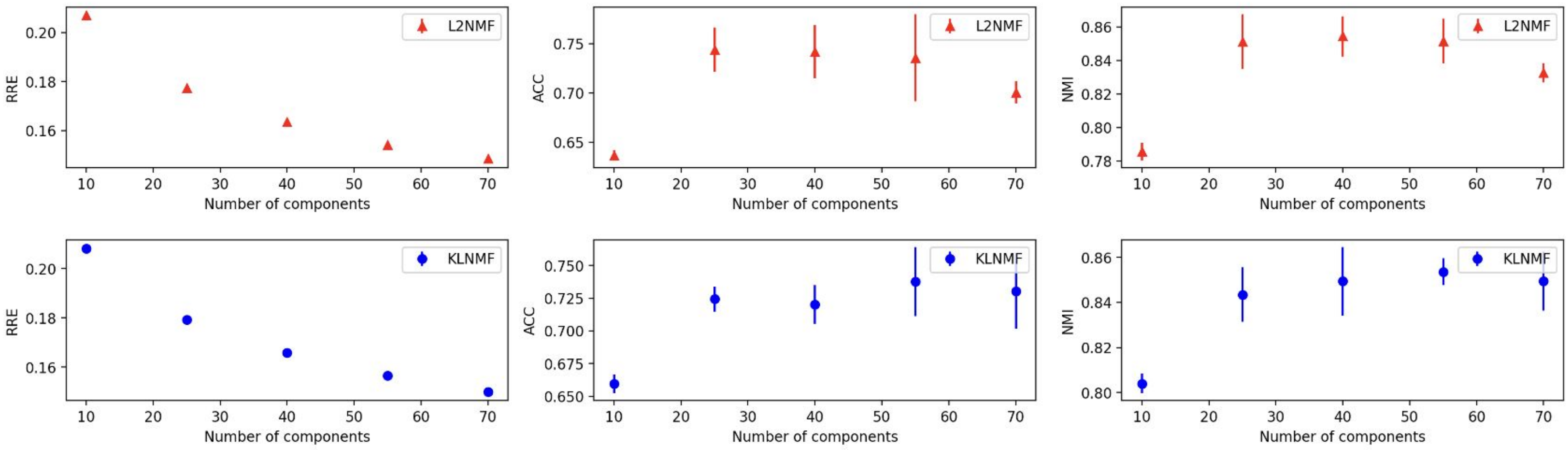


Figure 7: Mean and standard deviation of multiple runs

k = 60 is good

Influence of Noise

- Apply Salt&Pepper, Lapacian, Gaussian noise to images
- Set no. components k = 60 as found previously
- Document the mean and sd of multiple runs
- Experimental setup:

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

Table 2: Table of parameters used in the experiments

Influence of Noise

- Apply Salt&Pepper, Lapacian, Gaussian noise to images
- Gaussian noise

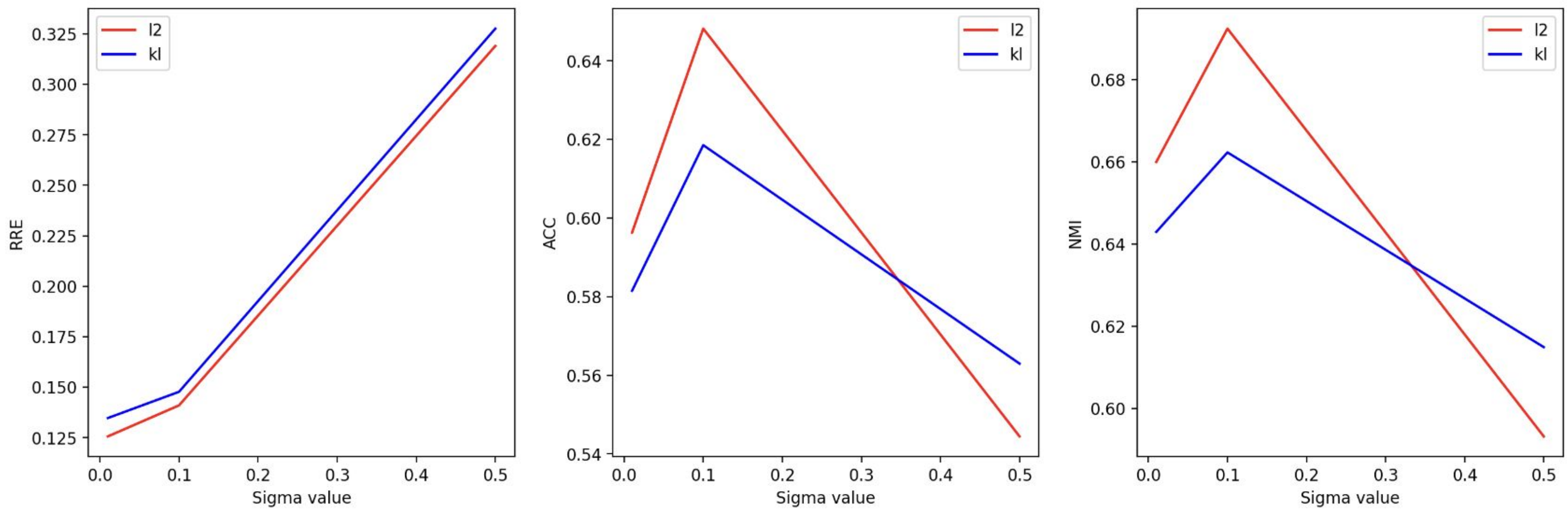


Figure 8: Influence of σ value on REE, ACC and NMI

Influence of Noise

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

Table 3: Table of results for Noise experiment

Conclusion

- Investigate and discuss the convergence of KL-NMF algorithm

- Present and implement the formulation of 2 types of NMF

- Experiment with SVM shows NMF is a Speed/Interpretability - Performance tradeoff

- Examine the effect of varying no. components k shows the performance reach a maximum then decline

- Implement 3 types of noise and analyze the interactions between them and the 2 NMF algorithms

Future work

Algorithms:

- Continue to investigate the convergence of NMF

Experiments:

- Experiment with more data
- Try other face databases, also work with colored images
- NMF in other tasks rather than Facial recognition

    e.g: signal processing, topic modeling,....

References

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

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

[3] Lee, DD & Seung, HS. Learning the parts of objects by non-negative matrix factoriza- tion. Nature 401, 788- 791. 1999

[4] Lee, DD & Seung, HS. Algorithms for non-negative matrix factorization. In Proceedings of the 13th International Conference on Neural Information Processing Systems (NIPS'00). MIT Press, Cambridge, MA, USA, 535–541. 2000