EEE-361 HOMEWORK 1

February 15, 2021

The purpose of this homework is to compare matrix factorization techniques through a facial recognition study. For this purpose, the CBCL database, originally used in [3], which is made available in Moodle will be used. Note that the dataset is divided into train and test datasets. In order to perform your first task, you will use the train dataset. The techniques from [1] will be used in order to carry out the comparison. Proceed as described in the following steps and report your findings and results as required. Note that the required programming can be implemented in MATLAB or Python.

- Download the facial database from your Moodle account.
- Describe how the matrix $\boldsymbol{X} \in \mathbb{R}^{p \times n}$ (see [1]) should be formed.

1 Implementation and analysis

The following steps correspond to the comparison between Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF).

1. Singular Value Decomposition (SVD)
   (a) Use SVD to factorize $\boldsymbol{X}$ as
   $$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T ,$$
   where $\boldsymbol{\Sigma}$ is a 361×361 square matrix with diagonal entries σ_i^2, $1 \leq i \leq 361 = 19 \times 19$.
   (b) Plot the singular values as $\mathsf{plot}([1{:}361], \mathsf{diag}(\boldsymbol{\Sigma}))$. Then compute the accumulated energy $\boldsymbol{e}$ of the singular values as: $\boldsymbol{e}[i] = \sum_{k=1}^{i} \sigma_k^2$, for $1 \leq i \leq 361$. Afterwards, plot the normalized accumulated energy as $\mathsf{plot}([1{:}361], \boldsymbol{e}/\mathsf{max}(\boldsymbol{e}))$, where $\mathsf{max}(\boldsymbol{e})$ represents the maximum element of the vector $\boldsymbol{e}$.
   (c) Identify the indices I_{90}, I_{95} and I_{99} of $\boldsymbol{e}$ that correspond to the indices when the normalized energy reaches to 0.9, 0.95 and 0.99, respectively.
   (d) Check the corresponding singular faces which are the first I_{90} columns of $\boldsymbol{U}$ and display them as in Figure 1 of [1]. Comment on the obtained singular faces? Do they have localized or distributed features? Are they non-negative valued?
2. Non-negative Matrix Factorization (NMF)
   (a) Use the Non-negative Matrix Factorization (NMF) technique called HALS in Section 3.1.5 of [1] to factorize $\boldsymbol{X} = \boldsymbol{W}\boldsymbol{H}$, where both $\boldsymbol{W} \in \mathbb{R}^{361 \times r}$ and $\boldsymbol{H} \in \mathbb{R}^{r \times 2429}$ have non-negative entries, for some $r \in \mathbb{N}$ to be determined later. For this purpose do the following:
      - Write a code that implements the HALS update given in Section 3.1.5 of [1].
      - Initialize the algorithm by using the SVD-based initialization as described in Section 3.1.8 of [1]. Here you will use the decomposition that you found in the previous section in the following way. First, choose the value of r using SVD or by trial and error if the SVD choice does not seems satisfactory. Now let us define $[\boldsymbol{x}]_+ = \max(x, 0)$ and $[\boldsymbol{x}]_- = \max(-x, 0)$. Furthermore, for all $1 \leq k \leq r$, we define $[\boldsymbol{M}_k]_+ = [\boldsymbol{u}_k]_+ [\boldsymbol{v}_k^T]_+$, where $\boldsymbol{u}_k$ are columns of $\boldsymbol{U}$ and $\boldsymbol{v}_k$ are columns of $\boldsymbol{V}$. Similarly, we define $[\boldsymbol{M}_k]_- = [\boldsymbol{u}_k]_- [\boldsymbol{v}_k^T]_-$. If $\|[M]_+\|_F \geq \|[M]_-\|_F$, initialize the columns of $\boldsymbol{W}$ and rows of $\boldsymbol{H}$ as:
      $$\boldsymbol{w}_k = [\boldsymbol{u}_k]_+ / \|[\boldsymbol{u}_k]_+\| \quad \text{and} \quad \boldsymbol{h}_k^T = \sigma_k \|[\boldsymbol{u}_k]_+\| \, [\boldsymbol{v}_k]_+ .$$
      Otherwise, initialize them as:
      $$\boldsymbol{w}_k = [\boldsymbol{u}_k]_- / \|[\boldsymbol{u}_k]_-\| \quad \text{and} \quad \boldsymbol{h}_k^T = \sigma_k \|[\boldsymbol{u}_k]_-\| \, [\boldsymbol{v}_k]_- .$$
      - As a Stopping criterion of the HALS iterations you can use Section 4.2.2 of [2].
      - Plot the convergence of iterations as given in Figure 3 of [1]. Note that your plot will contain a single curve corresponding to the HALS results. Compare these results with the corresponding ones given in Figure 3 of [1].
   (b) Check the eigenfaces in $\boldsymbol{W}$ and display them as images as in Figure 1 of [1]. Comment on the obtained eigenfaces? Do they have localized or distributed features? Are they non-negative valued?

2 Image recovery from noisy data

Your next task is to accurately reconstruct images from their noisy versions. For this, the test dataset will be used. Let us denote by $\boldsymbol{Y}_k \in \mathbb{R}^{19\times 19}$ the kth image from the test dataset, where $1 \leq k \leq 472$. First, you need to add random noise to this image in order to obtain $\widetilde{\boldsymbol{Y}}_k$, where the (i,j) entry of $\widetilde{\boldsymbol{Y}}_k$ is defined as

$$\widetilde{\boldsymbol{Y}}_k(i,j) = \min\{\boldsymbol{Y}_k + \eta \cdot \mathsf{rand}, 255\},$$

where η is a scaling parameter. You need to perform the following task for the values $\eta = 1, 10, 25$ and comment on the results.

Then, the noisy image should be vectorized. In order to do that, let us formally define the following function:

$$\begin{aligned} \mathsf{vec}: \ & \mathbb{R}^{m\times n} \longrightarrow \mathbb{R}^{mn} \\ & \begin{bmatrix} & & \\ \boldsymbol{v}_1 & \dots & \boldsymbol{v}_n \\ & & \end{bmatrix} \mapsto \begin{bmatrix} \boldsymbol{v}_1 \\ \vdots \\ \boldsymbol{v}_n \end{bmatrix}, \end{aligned}$$

where $\boldsymbol{v}_i \in \mathbb{R}^m$, $1 \leq i \leq n$ are the columns of the input matrix. Now let us denote by $\boldsymbol{y}_k$ the vectorized version of $\boldsymbol{Y}_k$, that is, $\boldsymbol{y}_k = \mathsf{vec}(\boldsymbol{Y}_k)$. Similarly, we have $\widetilde{\boldsymbol{y}}_k = \mathsf{vec}(\widetilde{\boldsymbol{Y}}_k)$.

First, you should reconstruct the original image by using SVD. Recall the definition of I_{90} in the previous section. Let us denote it by r_{SVD} from here on. You will use the first r_{SVD} singular faces from the matrix $\boldsymbol{U}$ obtained above in order to reconstruct the original images. What we have so far are the noisy vectorized images $\widetilde{\boldsymbol{y}}_k$, $1 \leq k \leq 472$. For each $\widetilde{\boldsymbol{y}}_k$, we denote by $\overline{\boldsymbol{y}}_k^{SVD}$ its reconstructed version. That is:

$$\overline{\boldsymbol{y}}_k^{SVD} = \sum_{p=1}^{r_{SVD}} \alpha_p \boldsymbol{u}_p \ ,$$

where $\alpha_p = \widetilde{\boldsymbol{y}}_k^T \boldsymbol{u}_p$ is the dot product of $\widetilde{\boldsymbol{y}}_k$ and $\boldsymbol{u}_p$ and $\boldsymbol{u}_p$ are the singular faces of $\boldsymbol{X}$ (columns of $\boldsymbol{U}$), for every $1 \leq p \leq r_{SVD}$.

Similarly, you should reconstruct the original image by using NMF. Denote by r_{NMF} the value of r that you chose for its implementation. Therefore, we have the weight matrix $\boldsymbol{W} \in \mathbb{R}^{361\times r_{NMF}}$. The reconstructed vector $\overline{\boldsymbol{y}}_k^{NMF}$ will in this case be:

$$\overline{\boldsymbol{y}}_k^{NMF} = \sum_{p=1}^{r_{NMF}} \beta_p \boldsymbol{w}_p \ ,$$

where $\boldsymbol{w}_p$ are the eigenfaces of $\boldsymbol{X}$ (columns of $\boldsymbol{W}$) and $\boldsymbol{\beta} = (\boldsymbol{W}^T\boldsymbol{W})^{-1}\boldsymbol{W}^T\widetilde{\boldsymbol{y}}_k$ is the least squares solution of $\boldsymbol{W}$ corresponding to $\widetilde{\boldsymbol{y}}_k$; note that β_p is, as the pth element of $\boldsymbol{\beta}$, a scalar.

Next, we transform $\overline{\boldsymbol{y}}_k^{SVD}$ and $\overline{\boldsymbol{y}}_k^{NMF}$ back to a matrix via the following function:

$$\begin{aligned} \mathsf{mat}: \ & \mathbb{R}^{mn} \times \mathbb{R} \longrightarrow \mathbb{R}^{m\times n} \\ & \left(\begin{bmatrix} \boldsymbol{v}_1 \\ \vdots \\ \boldsymbol{v}_n \end{bmatrix}, \ m \right) \mapsto \begin{bmatrix} & & \\ \boldsymbol{v}_1 & \dots & \boldsymbol{v}_n \\ & & \end{bmatrix}. \end{aligned}$$

Note that mat takes as input the vector and the the size m of the columns of the output matrix. We denote the SVD-reconstructed image by $\overline{\boldsymbol{Y}}_k^{SVD}$, i.e. $\overline{\boldsymbol{Y}}_k^{SVD} = \mathsf{mat}(\overline{\boldsymbol{y}}_k^{SVD}, 19)$, and the NMF-reconstructed image by $\overline{\boldsymbol{Y}}_k^{NMF}$, i.e., $\overline{\boldsymbol{Y}}_k^{NMF} = \mathsf{mat}(\overline{\boldsymbol{y}}_k^{NMF}, 19)$ since we want a 19×19 image. The following diagrams describe the whole procedure for the SVD-based and NMF-based reconstruction respectively.

$$\boldsymbol{Y}_k \xrightarrow{+\mathsf{rand}(19,19)} \widetilde{\boldsymbol{Y}}_k \xrightarrow{\mathsf{vec}(\widetilde{\boldsymbol{Y}}_k)} \widetilde{\boldsymbol{y}}_k \xrightarrow{SVD} \overline{\boldsymbol{y}}_k^{SVD} \xrightarrow{\mathsf{mat}(\overline{\boldsymbol{y}}_k^{SVD},19)} \overline{\boldsymbol{Y}}_k^{SVD} \ ;$$

$$\boldsymbol{Y}_k \xrightarrow{+\mathsf{rand}(19,19)} \widetilde{\boldsymbol{Y}}_k \xrightarrow{\mathsf{vec}(\widetilde{\boldsymbol{Y}}_k)} \widetilde{\boldsymbol{y}}_k \xrightarrow{NMF} \overline{\boldsymbol{y}}_k^{NMF} \xrightarrow{\mathsf{mat}(\overline{\boldsymbol{y}}_k^{NMF},19)} \overline{\boldsymbol{Y}}_k^{NMF} .$$

For every reconstructed image $\overline{\boldsymbol{Y}}_k^{SVD}$ and $\overline{\boldsymbol{Y}}_k^{NMF}$, you will compute the Frobenius norm of the corresponding error $\boldsymbol{E}_k^{SVD}$ and $\boldsymbol{E}_k^{NMF}$ respectively, where the error is defined as $\boldsymbol{E}_k = \boldsymbol{Y}_k - \overline{\boldsymbol{Y}}_k$. So for each k, the norm $\left\|\boldsymbol{E}_k^{SVD}\right\|_F$ and $\left\|\boldsymbol{E}_k^{NMF}\right\|_F$ should be computed. The final step is to plot the average of the errors over the 472 test images versus different values of r_{SVD} and r_{NMF}. So the y-axis of the first graph contains the values of

$$\mathsf{error}(r_{SVD}) := \frac{1}{472}\sum_{k=1}^{472} \left\|\boldsymbol{E}_k^{SVD}\right\|_F$$

corresponding to different values of r_{SVD} in the x-axis. In the second graph you will show the values of $\mathsf{error}(r_{NMF})$ for different values of r_{NMF} similarly.

Comment on the results and compare. What can you say about the change of the average error as r_{SVD} and r_{NMF} change? Is there a trend?

3 Image recovery from masked data

For the next task, you need to recover images from their masked versions. A masked image is the entry-wise product of the original image matrix with a shadow mask matrix $\boldsymbol{S}$. For this homework, the (i, j) entry of the matrix $\boldsymbol{S}$ is defined as follows:

$$\boldsymbol{S}(i,j) = \begin{cases} 1 & \text{if } j \leq 10 \\ 1 - 0.05(j-10) & \text{if } j > 10 \end{cases}$$

Using the same notation as in the previous section we now have

$$\widetilde{\boldsymbol{Y}}_k = \boldsymbol{Y}_k \odot \boldsymbol{S},$$

where we denote by $\odot$ the entry-wise product, that is, $\widetilde{\boldsymbol{Y}}_k(i,j) = \boldsymbol{Y}_k(i,j) \cdot \boldsymbol{S}(i,j)$, for all $1 \leq i, j \leq 19$. Next you should proceed as in the previous section in order to obtain $\overline{\boldsymbol{Y}}_k^{SVD}$ and $\overline{\boldsymbol{Y}}_k^{NMF}$ corresponding to both methods. Note that in this case, the diagrams will look as follows:

$$\boldsymbol{Y}_k \xrightarrow{\odot \boldsymbol{S}} \widetilde{\boldsymbol{Y}}_k \xrightarrow{\mathsf{vec}(\widetilde{\boldsymbol{Y}}_k)} \widetilde{\boldsymbol{y}}_k \xrightarrow{SVD} \overline{\boldsymbol{y}}_k^{SVD} \xrightarrow{\mathsf{mat}(\overline{\boldsymbol{y}}_k^{SVD},19)} \overline{\boldsymbol{Y}}_k^{SVD} ;$$

$$\boldsymbol{Y}_k \xrightarrow{\odot \boldsymbol{S}} \widetilde{\boldsymbol{Y}}_k \xrightarrow{\mathsf{vec}(\widetilde{\boldsymbol{Y}}_k)} \widetilde{\boldsymbol{y}}_k \xrightarrow{NMF} \overline{\boldsymbol{y}}_k^{NMF} \xrightarrow{\mathsf{mat}(\overline{\boldsymbol{y}}_k^{NMF},19)} \overline{\boldsymbol{Y}}_k^{NMF} .$$

Finally, you will again plot two graphs showing the change in $\mathsf{error}(r_{SVD})$ and $\mathsf{error}(r_{NMF})$ for different values of r_{SVD} and r_{NMF} respectively.

Comment on the results and compare. What can you say about the change of the average error as r_{SVD} and r_{NMF} change? Is there a trend?

References

[1] N Gillis. The why and how of nonnegative matrix factorization, 2014. *arXiv Preprint arXiv:1401.5226v2*, 2014.

[2] Nicolas Gillis et al. Nonnegative matrix factorization: Complexity, algorithms and applications. *Unpublished doctoral dissertation, Université catholique de Louvain. Louvain-La-Neuve: CORE*, 2011.

[3] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.