Assignment 1—Group:

Chen Chen (123123123), Dania(123123123), Joyce(123123123)

17th September 2018

Abstract

Abstract text goes here, justified and in italics. The abstract would normally be one paragraph long.

Contents

1 Introduction

Non-negative matrix factorization (NMF) is a matrix decomposition technique that approximates a multivariate data matrix by two lower dimensional non-negative matrices as follows:

$$\nu\nu \tag{1}$$

As NMF only allows additive, non-subtractive combination of matrix factors, it is applicable to an extensive range of domain. Lee and Seung [1999] suggest that NMF is useful for image processing problems including facial recognition. Specifically, NMF generates two matrices W and H which are often referred as the basis images and weights. This is because the observed image V is approximated by a linear combination of W and H. This property also distinguishes NMF from other traditional image processing methods such as principal components analysis (PCA) and K-means clustering. Guillamet and Vitrià [2002] demonstrate that NMF is more robust to corrupted images than PCA. Moreover, NMF is also applicable to text mining such as semantic analysis. More generally, NMF is useful to discover semantic features of an article by counting the frequency of each word than approximating the document from a subset of a large array of features [Lee and Seung, 1999].

Researchers proposed many NMF algorithms. Lee and Seung [2001] first proposes "multiplicative update rules" to minimise Euclidean distance or Kullback-Leibler divergence between the original matrix and its approximation. Although this algorithm is easy to implement and have reasonable convergent rate [Lee and Seung, 2001], it may fail on seriously corrupted dataset which violates its assumption of Gaussian noise or Poisson noise, respectively [Guan et al., 2017]. To improve the robustness of NMF, many methods have been proposed. **?** proposes L_1-norm based NMF to model noisy data by a Laplace distribution which is less sensitive to outliers. However, as L_1-norm is not differentiable at zero, the optimization procedure is computationally expensive. Kong et al. [2011] proposed an NMF algorithm using L_{21}-norm loss function which is robust to outliers. The updating rules used in L_{21}-norm NMF, however, converge slowly because of a continual use of the power method [Guan et al., 2017].

In practice, face images could be easily corrupted during data collection by large magnitude noise. Corruption may result from lighting environment, facial expression or facial details. An NMF algorithm that is robust to large noise is desired for real-world application. Therefore, the objective of this project is to analyse the robustness of NMF algorithms on corrupted dataset. We implement two NMF algorithms on real face image datasets, ORL dataset and Extended YaleB dataset. The face images are contaminated by artificial noises.

Plan: ...

2 Related work

DiscussionDiscussionDiscussion

3 Methods

3.1 NMF and Gaussian noise

Lee and Seung [2001] propose the first NMF with the objective function between imaga V and its NMF factorisation W and H being

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

To minimise this object function of least square, Lee and Seung [2001] prove the convergence of the multiplication update rule

$$H_{jk} \leftarrow H_{jk} \frac{(W^T V)_{jk}}{(W^T WH)_{jk}} \text{ and } W_{ij} \leftarrow W_{ij} \frac{(VH)_{ij}}{(WHH^H)_{ij}}. \tag{3}$$

Our lecturer derived this NMF algorithm minimises Gaussian noise in the lecture.

3.2 KLNMF and Poisson noise

Lee and Seung [2001] suggest that KLNMF is a algorithm that minimising the Kullback-Leibler divergence

$$D(V||WH) = \sum_{ij}\left(V_{ij}\log\frac{V_{ij}}{(WH)_{ij}} - V_{ij} + (WH)_{ij}\right) \quad (4)$$

$$= \sum_{ij}\left(-V_{ij}\log(WH)_{ij} + (WH)_{ij} + C(V_{ij})\right). \quad (5)$$

where $C(V_{ij}) = V_{ij}\log V_{ij} - V_{ij}$. $C(V_{ij})$ is a function of the observed image matrix V only. Lee and Seung [2001] also suggest a multiplication update rule to find as the optimisation procedure of KLNMF

$$H_{jk} \leftarrow H_{jk}\frac{\sum_i W_{ij}V_{ik}/(WH)_{jk}}{\sum_{i'} W_{i'j}} \text{ and } W_{ij} \leftarrow W_{ij}\frac{\sum_k H_{jk}V_{ik}/(WH)_{jk}}{\sum_{k'} H_{ik'}}. \quad (6)$$

As this original image matrix V is observed, minimising this Kullback-Leibler divergence (5) is equivalent to minimising

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

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

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

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

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

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

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

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

3.3 Asymptotic equivalence of noise distributions

We design an Gaussian noise and a Poisson noise with different magnitude. Poisson distribution with parameter λ (integer) is equivalent to the sum of λ Poisson distributions with parameter 1 [Walck, 1996, p. 45]. Hence for λ large, Central Limit Theorem implies that Poisson distribution with parameter λ is well approximated by $N(\lambda, \lambda)$. When applying Poisson noise to an image, we do not degree of freedom to choose any parameter. The variance is the magnitude of the pixels. To compare the robustness of KLNMF with NMF with different noise, we choose the variance of Gaussian noise to be the different from the magnitude of the pixel, that is, $N(0, \text{Var}) \neq N(0, V) \approx \text{Poi}(V) - V$. Figure 1 visualises the similarity of Poisson distribution and Normal distribution with parameter $V = 40$.

3.4 Preprocessing

We apply global centring and local centring to preprocess the image data `Vhat`

Algorithm 1: Centring image data

```
n_samples = len(Vhat)
# global centering
Vhat = Vhat - Vhat.mean(axis=0)
# local centering
Vhat -= Vhat.mean(axis=1).reshape(n_samples, -1)
Vhat -= Vhat.min()
```

4 Experiments

123

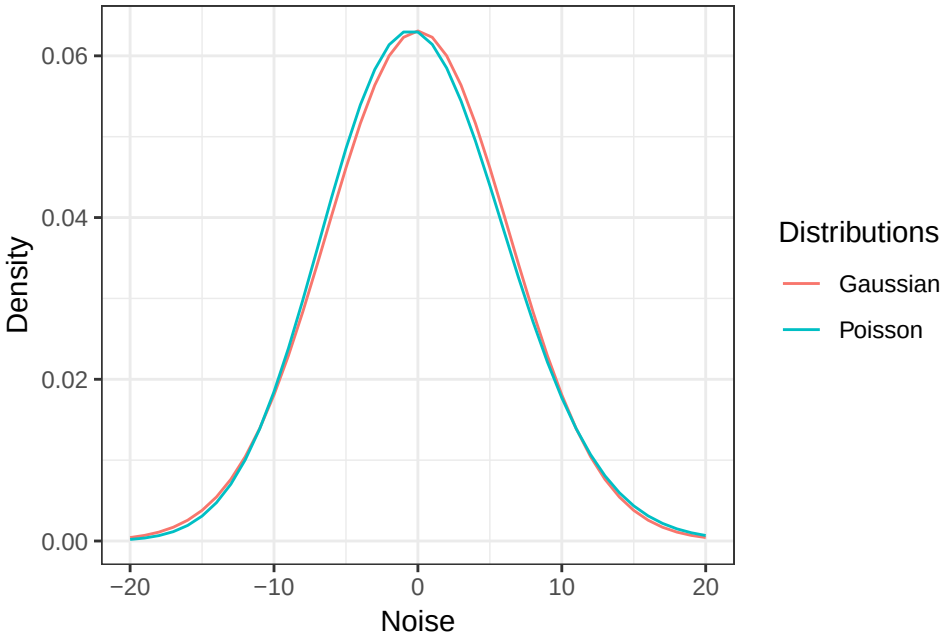

Figure 1: Compare a Gaussian noise $\mathsf{N}(0, 40)$ with Poisson noise $\mathrm{Poi}(40) - 40$. They two distributions are asymptotically equivalent and have similar density functions.

5 Conclusion

Your conclusion goes at the end, followed by References, which must follow the Vancouver Style (see: www.icmje.org/index.html). References begin below with a header that is centered. Only the first word of an article title is capitalized in the References.

A Appendix

Algorithm 2: The Levenberg–Marquardt algorithm iteratively finds optimal macroscale Robin boundary conditions.

```
%A=imread('C:\Users\chenc\OneDrive - UNSW\machine
    learning\assignment
    1\data\CroppedYaleB\yaleB01\yaleB01_P00A+000E+00.pgm');
red=3;
k=40;

imagefiles = dir('data/ORL/*/*.pgm');
imagefiles2=struct2cell(imagefiles);
imagefiles=imagefiles((~endsWith(imagefiles2(1,:),'Ambient.pgm'))');
imagefiles2=struct2cell(imagefiles);
A=imread(strcat(imagefiles(1).folder,'\',imagefiles(1).name));
if size(A,1)==112
    A=A(1:111,1:90);
end
A_list=zeros(size(A,1)/red,size(A,2)/red,red);
nfiles = length(imagefiles);    % Number of files found
matrix_image=zeros(prod(size(A))/red^2,nfiles);
temp=struct2cell(imagefiles);
names=temp(1,:)
for ii=1:nfiles
   currentfilename = imagefiles(ii).name;
   currentfilename=strcat(imagefiles(ii).folder,'\',currentfilename);
   currentimage = imread(currentfilename);
   if abs(size(A,1)-112)<=1
    currentimage=currentimage(1:111,1:90);
   end

   for i=1:red
    A_list(:,:,i)=currentimage(i:red:end,i:red:end);
   end
   A2=uint8(mean(A_list,3));
   matrix_image(:,ii) = A2(:);
end

[w h]=NeNMF(matrix_image,k);
idx = kmeans(h',k)
Y_pred=zeros(size(matrix_image,2),1)
namess=str2mat(string(imagefiles2(2,:))')
namess=str2num(namess(:,end-1:end))
for ii=unique(idx)'
   ind= (idx==ii);
```

```
40      Y_pred(ind)=mode(namess(ind,:));
41  end
42  nmi(Y_pred,namess)
```