HOMEWORK 2

Huynh Huu Dinh[1]

[1] *Institute of Statistics & Department of Applied Mathematics, National Chung Hsing University*

June 6, 2019

Answer for question 2,4 and 5

We will use dictionary learning to solve the problem of image denoising. The method for this homework follows quite closely the K-SVD approach described in this paper:
E.Elad and M. Aharon, *Image Denoising Via Sparse and Redundant representations over Learned Dictionaries*, IEEE Trans. on Image Processing, Vol. 15, no. 12, pp, 3736-3745, 2006.
Step 1: We add the toolbox *functions* (Gabriel Peyre) and the dataset *Cartoon* (similarly fot *Texture*).

```
addpath('functions/');
addpath('Cartoon/');
```

Step 2: Loading all images and convert them into grayscale matrices (or grayscale images).

```
% nxn is the size of each image
n=256;
% M_I is matrix which is combined by all
    grayscale matrices
M_I=[];
% num is number of images in Cartoon
num=13;
% Algorithm for finding all grayscale
   matrices and then add them into M_I
for i=1:num
    if (i==1)|(i==7)
        M0=imread(sprintf('%i.jpg',i));
        M0=imresize(M0,[n n]);
        M0=im2double(M0);
        M_I=[M_I M0];
    else
        M0=imread(sprintf('%i.jpg',i));
        M0=rgb2gray(M0);
        M0=imresize(M0,[n n]);
        M0=im2double(M0);
        M_I=[M_I M0];
    end
end
```

The Figure 1 illustrates for some grayscale images

```
% M_E is combined by three grayscale
    matrices: 4, 5 and 6
M_E=M_I(1:n,(4-1)*n+1:6*n);
imshow(M_E)
```

Figure 1: *Some grayscale images of Cartoon dataset*

Step 3: We create noised images from Cartoon dataset. However, for easy presentation, we only implement for one image (e.g., '7.png' in Cartoon dataset).

```
% K is name of K-th image
K=7;
% Sigma is noise level;
sigma=0.1;
% M0 is grayscale matrix of K-th image
M0=M_I(1:n,(K-1)*n+1:K*n);
% M is the noised grayscale matrix of K-
    th image
M=M0+sigma*randn(n);
```

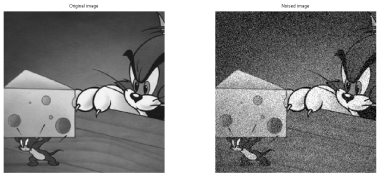

Figure 2: *Original and noised image*

```
figure;
subplot(1,2,1);imshow(M0);title('
    Original image');
subplot(1,2,2);imshow(M);title('Noised
    image');
```

Step 4: Since the learning is computationally intensive, we only apply it to small patches extracted from an image.

```
% Size of patches: w
w=12;
% Number of patches: m
m=40*w^2;
% Random patch location
x=floor(rand(1,1,m)*(n-w))+1;
y=floor(rand(1,1,m)*(n-w))+1;
% Extract lots of patches
[dY,dX]=meshgrid(0:(w-1),0:(w-1));
Xp=repmat(dX,[1 1 m])+repmat(x,[w w 1]);
Yp=repmat(dY,[1 1 m])+repmat(y,[w w 1]);
P=M(Xp+(Yp-1)*n);
```

We remove mean, since we are going to learn a dictionary of zero-mean and unit norm atom. The mean of the patches is close to being noise free.

```
P=P-repmat(mean(mean(P)),[w w]);
% Reshape so that each P(:,i)
    corresponds to a patch
P=reshape(P,[w^2 m]);
```

Display a few random patches.

```
plot_dictionary(P,[],[4 4]);
```

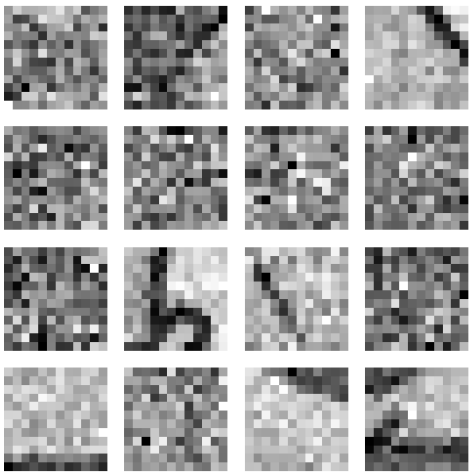

Figure 3: *Some random patches*

Step 5 (Sparse Coding): Given a dictionary D, we compute a set of coefficient X so that $D * X(:,i)$ is a sparse approximation of $P(:,i)$.

```
% Number of atoms in the dictionary
p=w^2;
% The initial dictionary is computed by
    a random selection of patches
sel=randperm(m);
sel=sel(1:p);
D=P(:,sel);
% Normalize the atoms
D=D./repmat(sqrt(sum(D.^2)),[w^2, 1]);
```

The sparse coding is obtained by minimizing a l_1 penalized optimization.

$$X(:,i) = \min_x 0.5 * norm(D * x - P(:,i))^2 + lambda * sum(abs(x))$$

This is achieved using an iterative thresholding method

$$x = thresh\_\{lambda * mu\}\,(x + mu * D' * (P\,(:,i) - D * x))$$

The value of $lambda$ controls the sparsity of the coefficients. Since l_1 regularization is similar to soft threshoding, we use the usual $3/2 * sigma$ value.

```
lambda=1.5*sigma;
% The gradient descent step size mu is
    related to the operator norm of the
    dictionary.
mu=1.9/norm(D)^2;
% Initialize the coefficients.
X=zeros(p,m);
% One step of iteration, for all the
    patches together
X=perform_thresholding(X+mu*D'*(P-D*X),
    lambda*mu);
```

Thus, X is sparse approximation of noised image. We save the matrix X into folder *Sparse Approximation*.

Step 6 (Automatic Set of the lambda Value): The value of $lambda$ is chosen arbitrary, and is the same for all the patches. Here we use an independent value of $lambda(i)$ for each patch $P(:,i)$.

```
lambda=zeros(m,1)+1.5*sigma;
```

We let $lambda(i)$ evolves during the optimization so that one has $norm(P(:,i) - D * X(:,i),'fro') = rho * w * sigma$, where rho is a damping factor close to 1. The value $w * sigma$ is approximately the amount of noise that contaminate $P(:,i)$.

```
rho=1.4;
error_target=rho*w*sigma;
```

we use the following update rule during the iterative thresholding.

```
lambda=lambda*error_target./sqrt(sum((P-
    D*X).^2))';
```

Step 7 (Update the Dictionary): Updating the dictionary is achieve by minimizing $norm(D * X - P,'fro')$ over all possible D. The solution by a pseudo-inverse.

```
D=P*pinv(X);
% The atoms are then normalized
D=D./repmat(sqrt(sum(D.^2)),[w^2,1]);
```

Display a few random patches of updated dictionary.

```
plot_dictionary(D,X,[4 4])
```

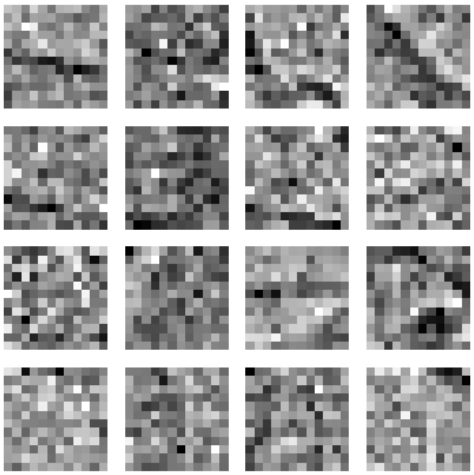

Figure 4: *Some random patches*

Step 8 (Denoising by Sparse Coding): The denoising of the image is obtained by sparse coding a large collection of patches (ideally all the patches).

```
% Overlap parameter (q=w is implies no
    overlap)
q=4;
% Regularly space positions for the
    extraction of patches
[y,x]=meshgrid(1:q:n-w/2,1:q:n-w/2);
m=size(x(:),1);
Xp=repmat(dX,[1 1 m])+repmat(reshape(x
    (:),[1 1 m]),[w w 1]);
Yp=repmat(dY,[1 1 m])+repmat(reshape(y
    (:),[1 1 m]),[w w 1]);
% Ensure boundary conditions
Xp(Xp>n)=2*n-Xp(Xp>n);
Yp(Yp>n)=2*n-Yp(Yp>n);
% Extract a large sub-set of regularly
    sampled patches.
P=M(Xp+(Yp-1)*n);
P=reshape(P,[w^2, m]);
% Save the mean of patches apart, and
    remove it.
a=mean(P);
P=P-repmat(a,[w^2 1]);
% Set a target error for denoising
rho=1;
error_target=rho*w*sigma;
% Update the dictionary again
sel=randperm(m);
sel=sel(1:p);
D=P(:,sel);
X=zeros(p,m);
X=perform_thresholding(X+mu*D'*(P-D*X),
    lambda*mu);
D=P*pinv(X);
D=D./repmat(sqrt(sum(D.^2)),[w^2,1]);
% Approximated patches
PA=reshape(D*X,[w w m]);
% Insert back the mean
PA=PA-repmat(mean(mean(PA)),[w w]);
PA=PA+reshape(repmat(a,[w^2 1]),[w w m])
    ;
% To obtain the denoising, we average
    the value of the approximated patches
% PA that overlap
W=zeros(n,n);
M1=zeros(n,n);
for i=1:m
    x=Xp(:,:,i);
    y=Yp(:,:,i);
    M1(x+(y-1)*n)=M1(x+(y-1)*n)+PA(:,:,i
        );
    W(x+(y-1)*n)=W(x+(y-1)*n)+1;
end
M1=M1./W;
```

Display the result

```
figure;
hold on
subplot(2,1,1);imshow(M1);title('
    Denoised');
subplot(2,1,2);imshow(M);title('Noising
    ');
```

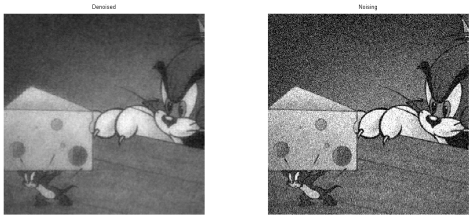

Figure 5: *Denoised and noised image*

In order to implement for all images, we run the file *Homework_2_Cartoon_all_image* and obtain figure 6.

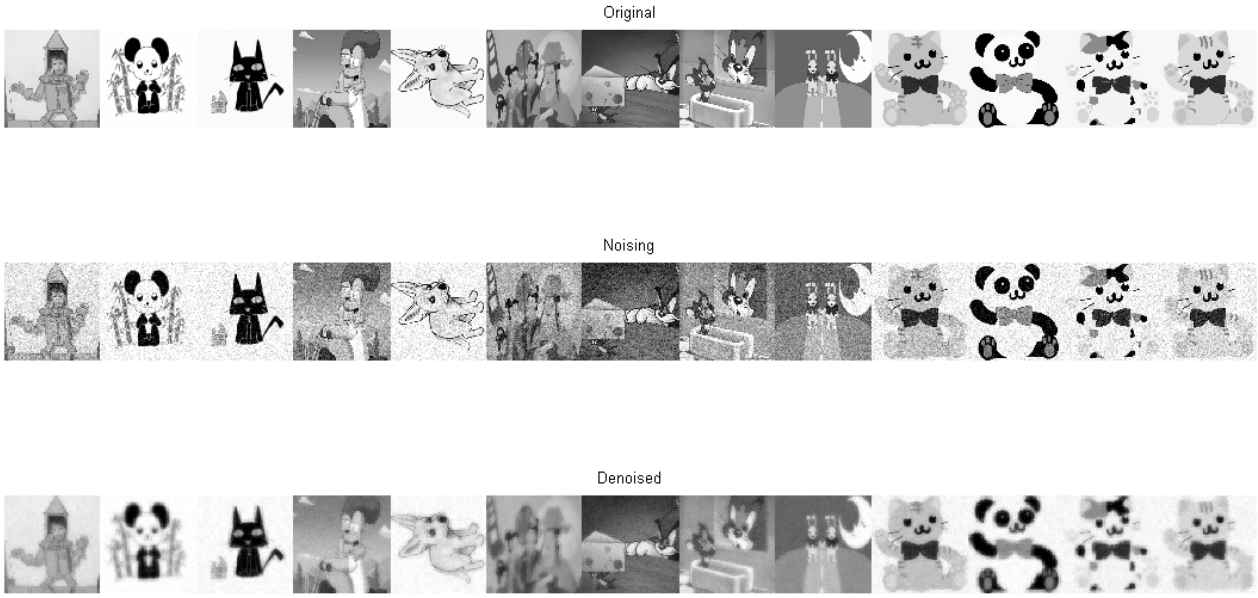

Figure 6: *All noised and denoised Cartoon images*

For Texture dataset, we run the file *Homework_2_Texture_all_image* and obtain figure 7.

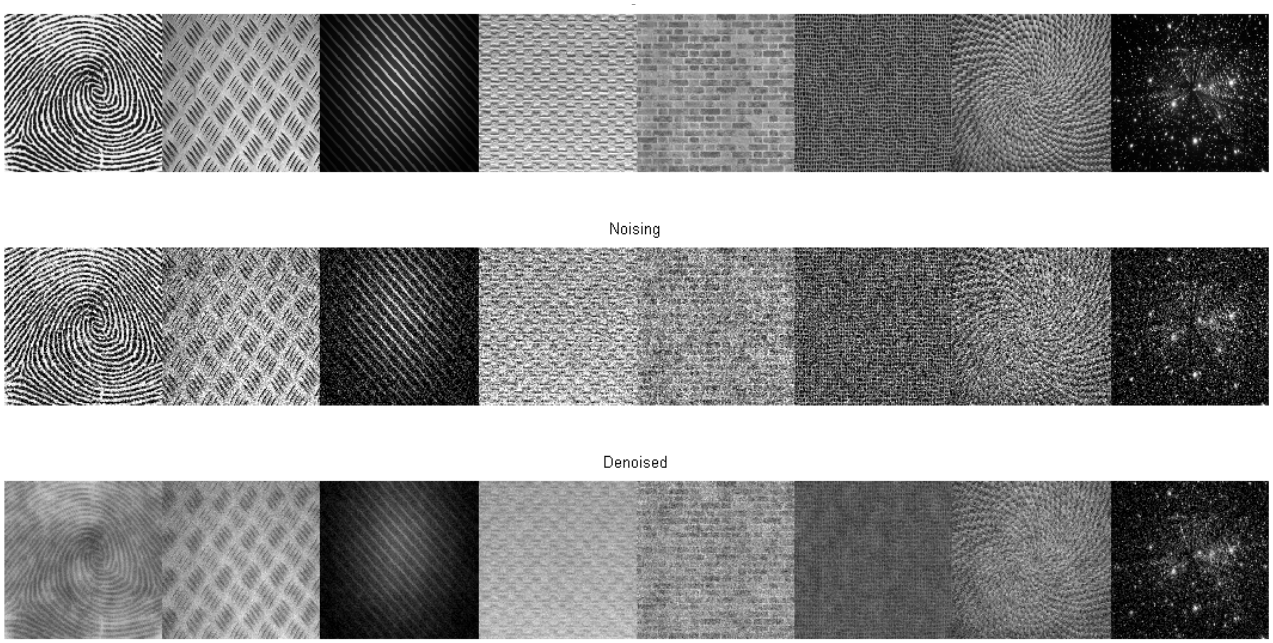

Figure 7: *All noised and denoised Texture images*