1. MATH547 HOMEWORK 2

Topic:	Math@UNC environment
Post date:	May 18, 2020
Due date:	May 19, 2020

1.1. Background

This homework investigates the matrix fundamental spaces, and the concept of linear dependence and independence.

1.2. Theoretical questions

1. Is the zero vector a linear combination of any non-empty set of vectors?

   Answer. Yes, $\mathscr{V} = (V, S, +, \cdot)$, $\boldsymbol{v} \in V$, $0 \in S$ since S a field, $0 \cdot \boldsymbol{v} = \mathbf{0}$, by vector space properties.

2. If $S \subseteq V$, with V a vector space then span(S) equals the intersection of all subspaces of V that contain S.

   Answer. Let I denote the intersection of all subspaces of V that contain S. Prove $\text{span}(S) = I$ by double inclusion:

   i. Consider $\boldsymbol{u} \in \text{span}(S)$, and let U be an arbitrary subspace of V that contains S. A subspace must be closed, hence $\boldsymbol{u} \in U$. Since $\boldsymbol{u}, U$ are arbitrary, $\text{span}(S) \subseteq I$.

   ii. Establish $I \subseteq \text{span}(S)$ by contradiction. Assume $\boldsymbol{u} \in I$, but $\boldsymbol{u} \notin \text{span}(S)$. If $\boldsymbol{u} \in I$, it must belong to all vector subpaces of V, including the $\text{span}(S)$ subspace, hence $\boldsymbol{u} \in \text{span}(S)$, contradiction. Conclude that if $\boldsymbol{u} \in I$ it follows that $\boldsymbol{u} \in \text{span}(S)$, hence $I \subseteq \text{span}(S)$.

3. Can a vector space have more than one basis?

   Answer. Yes. In $(\mathbb{R}, \mathbb{R}, +, \cdot)$ both $\{1\}$ and $\{-1\}$ are bases.

4. Must a vector space have a finite basis?

   Answer. No, the vector space of functions that can be differentiated an arbitrary number of times is not of finite dimension.

1.3. Fundamental spaces and linear dependence in image processing

1.3.1. Data input

Images are often processed using techniques from linear algebra. In this assignment a database of facial images from MIT will be used to investigate the fundamental vector subspaces associated with a matrix (linear mapping) and concepts of linear dependence. The database is available in a format readily loaded into Octave, with gray-scale images stored as column vectors of a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. There are n images, each with m pixels, and the two-dimensional image size in pixels is $p_x \times p_y$. The images have been pre-processed to remove non-uniform background illumination, noise, off-centering, different face size, and also been scaled such that the norm of each column vector is equal to one. The resulting images represent common facial features, but may seem distant from the much more detailed processing of visual information that leads a human to recognize a face. After loading the database, a directory is created for this assignment and set as the current directory

```
octave] cd /home/student/courses/MATH547ML/data/faces; load faces
octave] [m,n]=size(A); px=floor(sqrt(m)); py=m/px; disp([m n px py])
    16384      99     128     128
octave] cd /home/student/courses/MATH547ML;
octave] mkdir homework; cd homework; mkdir hw02; cd hw02
octave] pwd
ans = /home/student/courses/MATH547ML/homework/hw02
octave]
```

1.3.2. Utility functions

Define functions to display a facial image, and save an image to a file.

```
octave] function shwface(a,px,py)
          im=reshape(-a,px,py)'; colormap(gray);
          imagesc(im);
        end
octave] shwface(A(:,1),px,py)
octave] function savface(a,px,py,fname)
          im=reshape(-a,px,py)'; colormap(gray);
          imagesc(im); print(fname,"-deps");
        end
octave] savface(A(:,1),px,py,"face1");
octave] savface(A(:,2),px,py,"face2");
octave] savface(A(:,3),px,py,"face3");
octave]
```

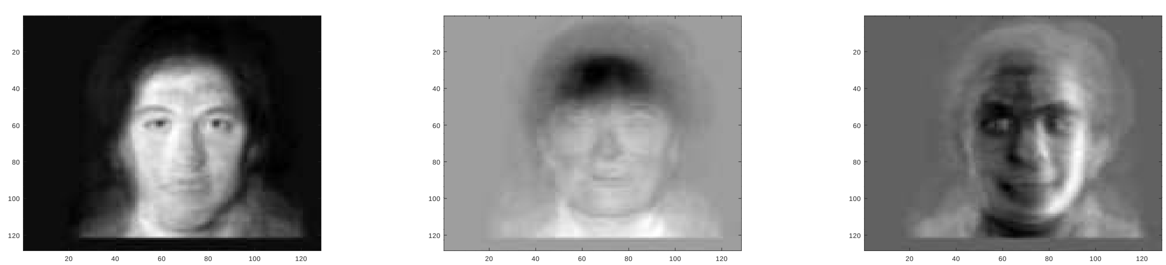

Figure 1.

1.3.3. Questions to investigate

Functionals to assess data variability. Define an average of the data $\boldsymbol{b} = \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{a}_j$, with $\boldsymbol{A} = [\ \boldsymbol{a}_1\ \ \boldsymbol{a}_2\ \ \dots\ \ \boldsymbol{a}_n\ ]$. Compute the norms $u_j = \|\boldsymbol{a}_j - \boldsymbol{b}\|_p$ and angle cosines $c_j = \boldsymbol{b}^T \boldsymbol{a}_j / \|\boldsymbol{b}\|$ (recall that preprocessing ensured $\|\boldsymbol{a}_j\| = 1$). Plot the results and comment on information provided by different norms. Assess data variability.

SOLUTION. Compute the average image

```
octave] b=mean(A')'; shwface(b,px,py);
octave] u=zeros(n,4);
octave] for i=1:n
          u(i,1) = norm(A(:,i)-b,1)/norm(b,1);
          u(i,2) = norm(A(:,i)-b,2)/norm(b,2);
          u(i,3) = norm(A(:,i)-b,3)/norm(b,3);
          u(i,4) = norm(A(:,i)-b,Inf)/norm(b,Inf);
        end
octave] idx=1:n; plot(idx,u(:,1),'k',idx,u(:,2),'r',idx,u(:,3),'b',idx,u(:,4),'g')
octave] print -depsc Q1Fig1.eps
octave] c = b'*A/norm(b);
octave] figure(2); plot(idx,c); print -depsc Q1Fig2.eps
octave]
```

In all norms $\|\boldsymbol{a}_j - \boldsymbol{b}\| \cong 10$ while $\|\boldsymbol{a}_j\| = 1$, so the deviation from the average is large. Sice $c_j \cong 0.1$ the cosine is almost constant. Deduce that the images are concentrated on the intersection of a hypercone of vertex angle $\theta \cong \arccos 0.1$ with a hypersphere of radius $r \cong 10$.

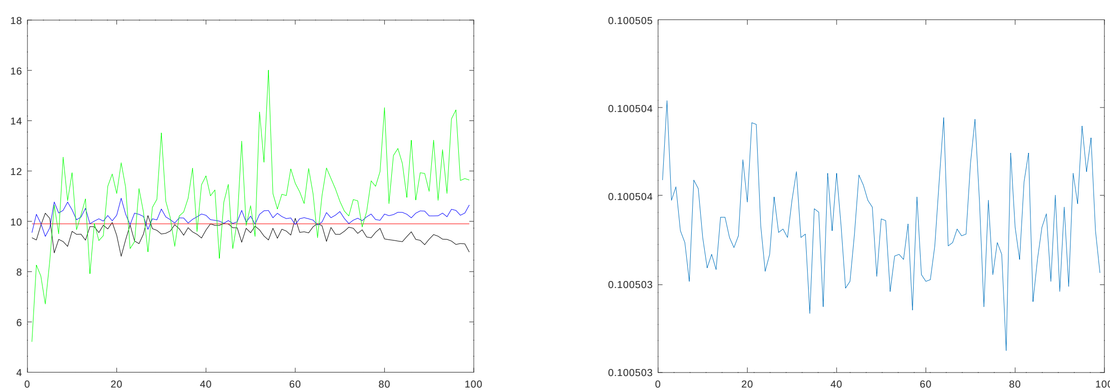

Figure 2.

Functional to assess data redundancy. If vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^m$ are colinear $\boldsymbol{x}^T \boldsymbol{y} / \|\boldsymbol{x}\| / \|\boldsymbol{y}\| = \pm 1$, one vector can be recovered from the other through a scaling operation $\boldsymbol{x} = a\boldsymbol{y}$, and the data is redundant. If the vectors are orthogonal, $\boldsymbol{x}^T \boldsymbol{y} = 0$, data in $\boldsymbol{x}$ is independent of data in $\boldsymbol{y}$. Assess whether facial data in $\boldsymbol{A}$ is independent by computing the angle cosines $s_{jk} = \boldsymbol{a}_j^T \boldsymbol{a}_k$.

SOLUTION. $\boldsymbol{S} = \boldsymbol{A}^T \boldsymbol{A}$ contains all the scalar products s_{jk}. Contour lines of $s(j,k) = \boldsymbol{a}_j^T \boldsymbol{a}_k$ (Fig. 3) shows that the column vectors of $\boldsymbol{A}$ are close to orthogonal since the off-diagonal elements are very small compared to the diagonal elements. hence there is no redundancy in the data.

```
octave] S=A'*A;
```

```
octave] figure(1); surf(S)
octave] figure(2); contour(S); print -depsc Q2Fig1.eps
octave] [S(10,10) S(10,20) S(10,30)]
ans =

   1.00000134838  -0.00000026211  -0.00000039673

octave] min(diag(S))
ans =  1.00000
octave]
```

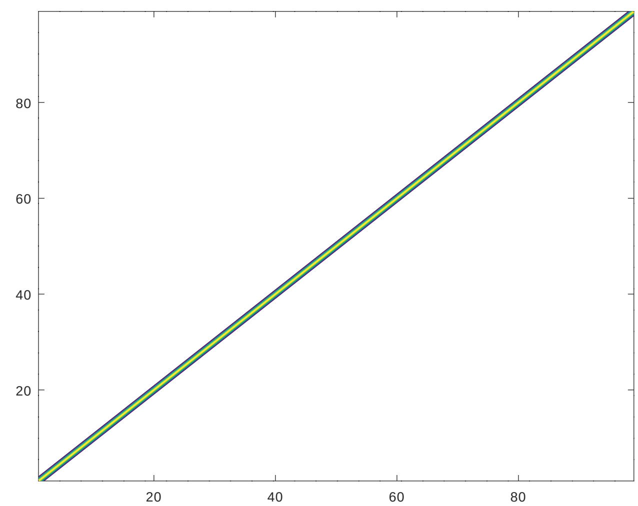

Figure 3. Scalar product of data vectors indicate linear independence; no redundancy

Column space to assess data sampling. The Octave `orth` function returns an orthognal set of vectors that span the column space of a matrix given as its argument. Display a few such vectors and comment on their utility by comparison to the column vectors of $\boldsymbol{A}$.

SOLUTION. Apply `orth` to the first 8 images in $\boldsymbol{A}$ to obtain $\boldsymbol{B}$, and display first 3 columns of each (Fig. 4). Both $\boldsymbol{A}, \boldsymbol{B}$ are equally useful in face identification since $\boldsymbol{A}$ was already close to orthogonal (previous question).

```
octave] B=orth(A(:,1:8));
octave] figure(1); shwface(A(:,1),128,128); print -depsc Q3Fig1a.eps
octave] figure(2); shwface(B(:,1),128,128); print -depsc Q3Fig1b.eps
octave] figure(1); shwface(A(:,2),128,128); print -depsc Q3Fig2a.eps
octave] figure(2); shwface(B(:,2),128,128); print -depsc Q3Fig2b.eps
octave] figure(1); shwface(A(:,3),128,128); print -depsc Q3Fig3a.eps
octave] figure(2); shwface(B(:,3),128,128); print -depsc Q3Fig3b.eps
```

```
octave]
```

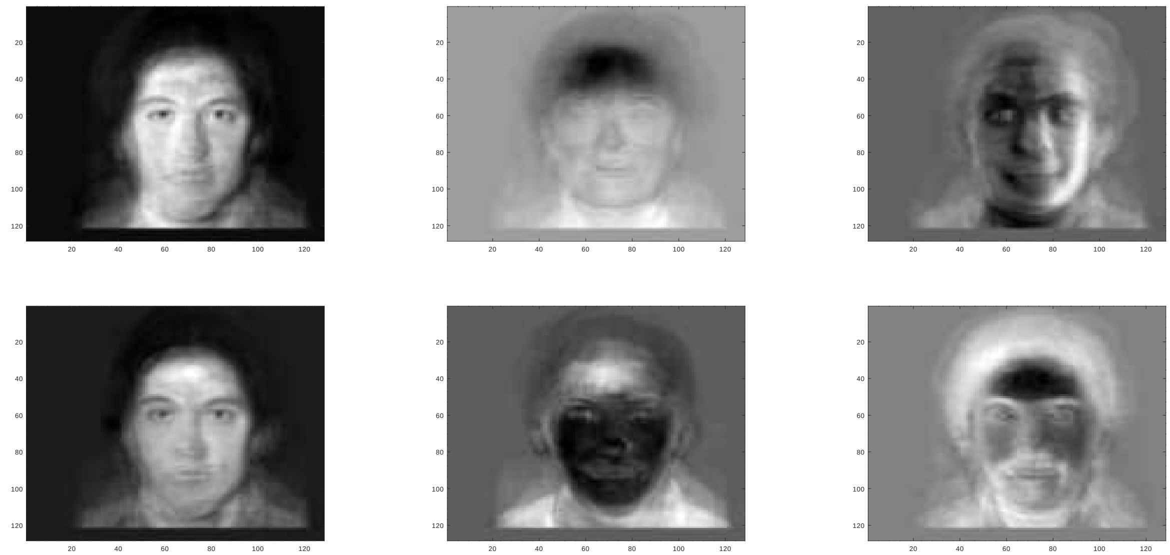

Figure 4. Images in $\boldsymbol{A}$ (top row), compared to images in $\boldsymbol{B}$ bottom row.

The same conclusion arises from sampling every other pixel in the horizontal and vertical directions to obtain 64×64 images, called subsampling (the shwface displays the image twice in this case)

```
octave] B=orth(A(1:4:m,1:8));
octave] figure(1); shwface(A(1:4:m,1),64,64); print -depsc Q3Fig4a.eps
octave] figure(2); shwface(B(:,1),64,64); print -depsc Q3Fig4b.eps
octave] figure(1); shwface(A(1:4:m,2),64,64); print -depsc Q3Fig5a.eps
octave] figure(2); shwface(B(:,2),64,64); print -depsc Q3Fig5b.eps
octave] figure(1); shwface(A(1:4:m,3),64,64); print -depsc Q3Fig6a.eps
octave] figure(2); shwface(B(:,3),64,64); print -depsc Q3Fig6b.eps
octave]
```

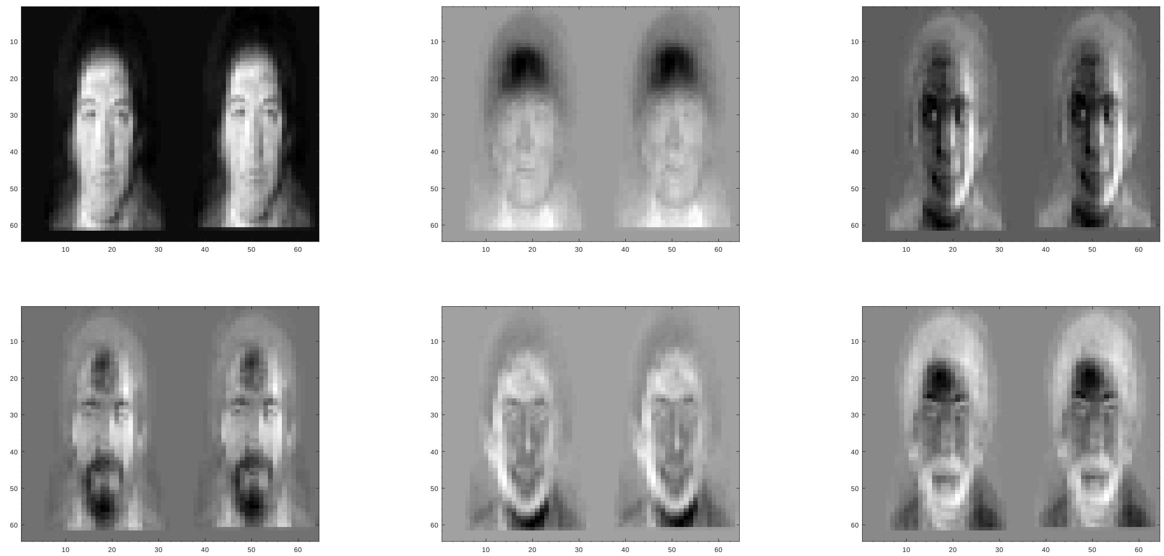

Figure 5. Images in $\boldsymbol{A}$ (top row), compared to images in $\boldsymbol{B}$ bottom row.

Left null space to assess missing data. The Octave `null` function returns an orthognal set of vectors that span the null space of a matrix given as its argument. Display a few vectors of $N(\boldsymbol{A}^T)$ and comment on what data cannot be obtained by linear combination of columns of $\boldsymbol{A}$.

SOLUTION. Compute $N(\boldsymbol{A}^T)$ on the 64×64 images obtained by subsampling, and display a few columns (Fig. 6). Notice that the vectors in the left null space contain isolated non-zero components. Linear combination of the facila images would not be able to represent such isolated pixels.

```
octave] N=null(A(1:4:m,1:8)');
octave] figure(1); shwface(N(:,1),64,64);  print -depsc Q4Fig1a.eps
octave] figure(1); shwface(N(:,31),64,64);  print -depsc Q4Fig1b.eps
octave] figure(1); shwface(N(:,61),64,64);  print -depsc Q4Fig1c.eps
octave]
```

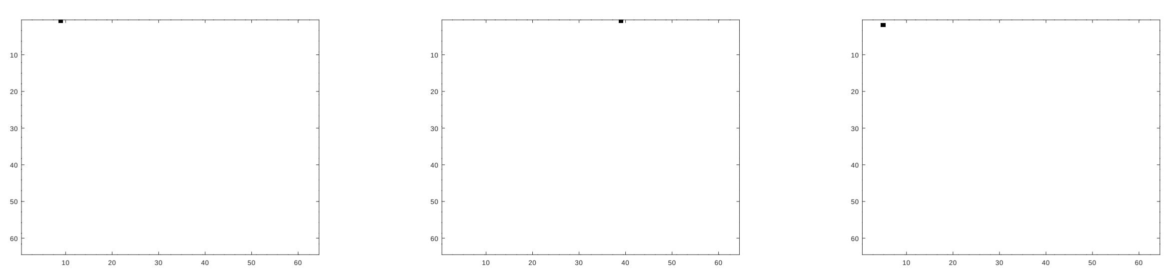

Figure 6. Elements of $N(\boldsymbol{A}^T)$

Linear combinations. Define a few new faces from linear combinations of $p = 2, 3, \ldots, 8$ columns of $\boldsymbol{A}$.

SOLUTION. Define scaling coefficients that sum to 1 for each p value, find the linear combination, display the image.

```
octave] x=[0.5; 0.5]; b=A(:,1:2)*x;
        shwface(b,128,128); print -depsc Q5Fig1.eps
```

```
octave] x=[0.2; 0.3; 0.5]; b=A(:,1:3)*x;
        shwface(b,128,128); print -depsc Q5Fig2.eps
octave] x=[0.1; 0.2; 0.2; 0.5]; b=A(:,1:4)*x;
        shwface(b,128,128); print -depsc Q5Fig3.eps
octave] x=[0.1; 0.2; 0.2; 0.2; 0.3]; b=A(:,1:5)*x;
        shwface(b,128,128); print -depsc Q5Fig4.eps
octave] x=[0.1; 0.2; 0.2; 0.2; 0.2; 0.1]; b=A(:,1:6)*x;
        shwface(b,128,128); print -depsc Q5Fig5.eps
octave] x=[0.1; 0.2; 0.2; 0.2; 0.1; 0.1; 0.1]; b=A(:,1:7)*x;
        shwface(b,128,128); print -depsc Q5Fig6.eps
octave] x=[0.1; 0.1; 0.1; 0.2; 0.2; 0.1; 0.1; 0.1]; b=A(:,1:8)*x;
        shwface(b,128,128); print -depsc Q5Fig7.eps
octave]
```

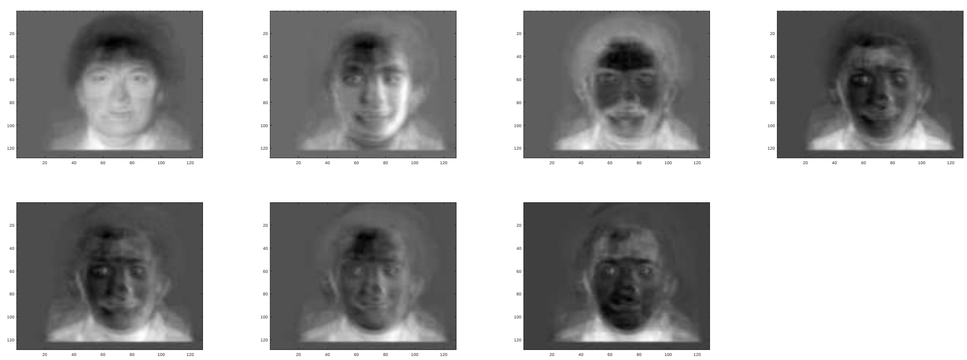

Figure 7.

Linear dependence. Is the facial data within $\boldsymbol{A}$ linearly dependent or independent?

SOLUTION. From the computation of the scalar products, $\boldsymbol{S} = \boldsymbol{A}^T \boldsymbol{A}$ with $\boldsymbol{S}$ close to diagonal, deduce that images within $\boldsymbol{A}$ are linearly independent.