Homework 1a

Please answer the following questions in complete sentences in a clearly prepared manuscript and submit the solution by the due date on Blackboard (around Friday, September 4th, 2015.)

Remember that this is a graduate class. There may be elements of the problem statements that require you to fill in appropriate assumptions. You are also responsible for determining what evidence to include. An answer alone is rarely sufficient, but neither is an overly verbose description required. Use your judgement to focus your discussion on the most interesting pieces. The answer to "should I include 'something' in my solution?" will almost always be: Yes, if you think it helps support your answer.

Problem 0: Homework checklist

- Please identify anyone, whether or not they are in the class, with whom you discussed your homework. This problem is worth 1 point, but on a multiplicative scale.
- Make sure you have included your source-code and prepared your solution according to the most recent Piazza note on homework submissions.

Problem 1: Operations

Compute the following by hand or using Matlab. The vector $\mathbf{e} = \begin{bmatrix}1 & \dots & 1\end{bmatrix}^T$ (i.e. the all ones vector).

1. $\begin{bmatrix}1 & 1 & 2\\ 3 & 5 & 8\\ 13 & 21 & 34\end{bmatrix}\begin{bmatrix}1 & -2 & 3\\ -4 & 5 & -6\\ 7 & -8 & 9\end{bmatrix} = \begin{bmatrix}11 & -13 & 15\\ 39 & -45 & 51\\ 167 & -193 & 219\end{bmatrix}$
2. $\mathbf{x} =$ `ones(1000,1)` $\mathbf{y} =$ `[1:1000]'` $\mathbf{x}^T\mathbf{y} = 500500.0$

   (Optional extra question – worth no points – who is always credited with discovering a very efficient way to compute this as a child?)

   Numpy users $\mathbf{x} =$ `ones((1000,1))` $\mathbf{y} =$ `arange(1.,1001.)[:,␣newaxis]'`
3. $\mathbf{x} = \begin{bmatrix}2 & 4 & -1\end{bmatrix}^T$.
   $\mathbf{e}\mathbf{x}^T$ =5
   $\mathbf{x}\mathbf{e}^T = \begin{bmatrix}2 & 2 & 2\\ 4 & 4 & 4\\ -1 & -1 & -1\end{bmatrix}$
4. $\mathbf{x} = \begin{bmatrix}1 & -18 & 3\end{bmatrix}^T$.
   $\mathbf{e}_1\mathbf{x}^T$ =-14
   $\mathbf{x}\mathbf{e}_3^T = \begin{bmatrix}1 & 1 & 1\\ -18 & -18 & -18\\ 3 & 3 & 3\end{bmatrix}$

Problem 2: A proof

Let $\boldsymbol{A}$ and $\boldsymbol{C}$ be invertible matrices. We'll prove that the inverse of $\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ 0 & \boldsymbol{C} \end{bmatrix}$ is easy to determine!

1. Show that the inverse of

   $$\begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}$$

   is

   $$\begin{bmatrix} 1 & -a \\ 0 & 1 \end{bmatrix}.$$

   Prove:

   $$\begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -a \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

2. Now, show that the inverse of

   $$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{A} \\ 0 & \boldsymbol{I} \end{bmatrix}$$

   is

   $$\begin{bmatrix} \boldsymbol{I} & -\boldsymbol{A} \\ 0 & \boldsymbol{I} \end{bmatrix}.$$

   Prove:

   $$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{A} \\ 0 & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{A} \\ 0 & \boldsymbol{I} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & 0 \\ 0 & \boldsymbol{I} \end{bmatrix}.$$

3. Recall that for general $\boldsymbol{A}$ and $\boldsymbol{B}$, not those in the problem!, $(\boldsymbol{AB})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1}$. Use this fact, and the result of problem 2.2 to determine the inverse to $\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ 0 & \boldsymbol{C} \end{bmatrix}$ when $\boldsymbol{A}$ and $\boldsymbol{C}$ are invertible. Alternatively, give the inverse of $\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ 0 & \boldsymbol{C} \end{bmatrix}$. Hint: think about diagonal matrices!

Problem 3: A statistical test

This problem will be more difficult if you haven't used Matlab or Scipy/Numpy before, so get started early! It's designed to give you some simple experience.

One way of viewing a rank-1 matrix is as any matrix that can be written as a single outer-product, that is, $\boldsymbol{A} = \mathbf{x}\mathbf{y}^T$ for some $\mathbf{x}$ and some $\mathbf{y}$.

In this problem, we'll look at the expected value of the *random matrix* $\boldsymbol{B}$ where $\boldsymbol{B}$ is a random rank-1 matrix following a simple probability distribution.

1. The first question is just to get you to document your initial impressions and is not worth any points. Suppose that $\mathbf{x}$ and $\mathbf{y}$ are length n vectors with elements drawn from a standard normal distribution. Let $\boldsymbol{B} = \mathbf{x}\mathbf{y}^T$. Then, let $\boldsymbol{C} = E[\boldsymbol{B}] = E[\mathbf{x}\mathbf{y}^T]$, where E is the expected-value operator. What do you think the rank of $\boldsymbol{C}$ is (guessing is fine!)? (Remember this isn't worth any points, so don't think about it too long or get hung up on small details.)

   The rank of $\boldsymbol{C}$ should be zero. Because for a standard normal distribution $\mathbf{x}$ the expectation value $E[x^2] = \sigma^2$

2. Now, let's check your guess! We can approximate $\boldsymbol{C}$ using what's called a Monte-Carlo approximation. Suppose that $\boldsymbol{B}_i = \mathbf{x}_i \mathbf{y}_i^T$ is a single matrix generated where $\mathbf{x}_i$ and $\mathbf{y}_i$ are *instances* of length-n vectors with elements drawn from a random normal. Then the Monte Carlo approximation to $\boldsymbol{C}$ is given by:

$$\boldsymbol{C} \approx \sum_{i=1}^{N} \frac{1}{N} \boldsymbol{B}_i = \sum_{i=1}^{N} \frac{1}{N} \mathbf{x}_i \mathbf{y}_i^T .$$

We can generate $\mathbf{x}_i$ and $\mathbf{y}_i$ in Matlab/Numpy/Julia via

Matlab

```
x = randn(n,1);
y = randn(n,1);
```

Numpy/Python

```
import numpy
x = numpy.random.randn(n,1) # Numpy
y = numpy.random.randn(n,1)
```

Julia

```
x = randn(n); # Julia
y = randn(n);
```

Using your language of choice, evaluate $\boldsymbol{C}$ where $N = 10000$ and $n = 6$ and report the rank.

3. Did you find the rank surprising given your initial guess? If so, comment on which you think is correct, your code (2) or your guess (1). (Hint: your code should be correct, but maybe you don't trust an *approximation*?)

Problem 4: Image downsampling

This problem will be more difficult if you haven't used Matlab or Scipy/Numpy before, so get started early! It's designed to teach you about writing `for` loops to construct a matrix operation for a particular task.

Consider the following problem. We have a 32×32 pixel image. Each pixel is a real-valued number between 0 and 1. However, I want to show this on a coffee maker screen and I only have a 16×16 pixel area to show the image. In order to reduce the size of the image, I want to average groups of 4 pixels. In this problem, we'll create a matrix-based program to do this

Let's work through a smaller example first. For the 4×4 image, represented here by a matrix-like array:

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{bmatrix}$$

I want to compute

$$\begin{bmatrix} (x_1 + x_2 + x_5 + x_6)/4 & (x_3 + x_4 + x_7 + x_8)/4 \\ (x_9 + x_{10} + x_{13} + x_{14})/4 & (x_{11} + x_{12} + x_{15} + x_{16})/4 \end{bmatrix}.$$

We will solve this problem using a matrix-vector product.

1. Suppose I call:

$$y_1 = (x_1 + x_2 + x_5 + x_6)/4 \qquad y_2 = (x_3 + x_4 + x_7 + x_8)/4$$
$$y_3 = (x_9 + x_{10} + x_{13} + x_{14})/4 \quad y_4 = (x_{11} + x_{12} + x_{15} + x_{16})/4.$$

Further, suppose we consider the vectors $\mathbf{y} \in \mathbb{R}^4$ and $\mathbf{x} \in \mathbb{R}^{16}$ to be the new image and old image, respectively. Write down the matrix $\boldsymbol{A}$ such that $\mathbf{y} = \boldsymbol{A}\mathbf{x}$.

In the remainder of the problem, we'll work through how to do this for a particular image in with additional guidance for using (i) Matlab or (ii) Scipy and Numpy.

You may use any software you like for this problem, although you must treat it as a matrix problem.

2. Download http://www.cs.purdue.edu/homes/dgleich/cs515-2014/homeworks/smallicon.mat and type

```
load smallicon.mat
```

in Matlab. You should get a 32×32 matrix $\boldsymbol{X}$. What is the sum of diagonal elements of $\boldsymbol{X}$? (If you wish to use another programming language, feel free, but make sure you can do the same things in it. I've provided http://www.cs.purdue.edu/homes/dgleich/cs515-2014/homeworks/smallicon.txt as a text file for use in other languages.)

Scipy and Numpy Use `scipy.io` to read Matlab mat files

```
from scipy.io import loadmat
filedata = loadmat('smallicon.mat')
X = filedata['X']
```

Julia Ask on Piazza for Julia help!

3. Matlab has a command for viewing a matrix as an image. Run the following commands:

```
imagesc(X)
```

The result looks really weird, right? That's because Matlab is making up a color for each pixel. We can tell it to use a greyscale colormap by executing:

```
colormap(gray)
```

Save the resulting Matlab figure as an image to include in your homework. (You'll need to do this for future homeworks, so make sure you know how to do it on this one!)

Scipy and Numpy use `imshow(X, extent=[0, 1, 0, 1])` and `set_cmap('gray')` instead

4. In what follows, we'll talk about two different types of indices. The image index of a pixel is a pair (i, j) that identifies a row and column for the pixel in the image. The vector index of a pixel is the index of that pixel in a linear ordering of the image elements. For instance, in the sample from part 1, pixel (3,2) has linear index 10. Also, pixel (1,4) has index 4. Matlab can help us built a map between pixel indices and linear or vector indices:

```
N = reshape(1:(4*4), 4, 4)';
```

This creates the pixel index to linear index for the problem above because

```
N(1,4)
N(3,2)
```

return the appropriate entry.

In your own words, explain what the `reshape` operation does. What happens if you omit the final transpose above and try:

```
N = reshape(1:(4*4), 4, 4);
```

instead?

Scipy and Numpy Beware that these index from 0 instead of 1, so you'll need to adjust the indices.

Julia Basically the same as Matlab.

5. Now we need to construct the matrix $\boldsymbol{A}$ in order to reduce the size of $\boldsymbol{X}$. Suppose we call the output vector $\mathbf{y}$ and the output image $\boldsymbol{Y}$. I'm giving you the following template, that I hope you can fill in. Feel free to construct $\boldsymbol{A}$ any way you choose, but the following should provide some guidance.

```
NX = <fill in>; % the map between pixel indices and linear indices for X
NY = <fill in>; % the map between pixel indices and linear indices for Y
for i=1:32
    for j=1:32
        xi = <fill in>; % the index of the pixel i,j in the vector x
        yij = <fill in>; % the resulting location of pixel in the matrix Y
        yi = <fill in>; % the index of the linear pixel in the vector y

        A(yi,xi) = 1/4; % place the entry of the matrix
    end
 end
```

6. In order to use the matrix $\boldsymbol{A}$ we created, we need to convert the matrix $\boldsymbol{X}$ into a vector! The reshape operation helps here:

```
x = reshape(X',32*32,1);
```

We can now rescale the image $\boldsymbol{X}$ by multiplying by $\boldsymbol{A}$ and reorganizing back into a matrix $\boldsymbol{Y}$.

```
y = A*x;
Y = reshape(y,16,16)';
```

Show the image of $\boldsymbol{Y}$. Does that look correct?

Scipy and Numpy Use `X.T` for the transpose of `X`, also make sure to use `dot(A,x)` for the matrix multiply.

Julia Like Matlab

7. The Matlab way to solve this problem is to use the built in routine `interp2`. In this case, you can call

```
Ym = interp2(X,-1)
```

to accomplish the same goal and save the output to a Matlab variable `Ym`.

Do `Y` and `Ym` agree? If not, can you comment on their differences?

Scipy and Numpy

```
from scipy.interpolate import RectBivariateSpline
Ym = RectBivariateSpline(
        arange(X.shape[0]),arange(X.shape[1]), A
     )(arange(0.,X.shape[0],2.),  arange(0.,X.shape[1],2.))
```

Julia Like Matlab

8. Show all of your code for this problem and document it.