CS/ECE/ME 532
Homework 5: matrix norms and the SVD

due: Friday October 21, 2016

1. Induced norms. In class, we defined the induced 2-norm of a matrix $A \in \mathbb{R}^{m\times n}$ as follows.

$$\|A\| = \max_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2}$$

Prove that the induced 2-norm is indeed a norm.

SOLUTION: We will verify each of the three properties of the norm individually.

- Proving that $\|A\| = 0$ if and only if $A = 0$:
  Proof: by the definition of $\|A\|$, it is clear that $\|A\| \geq 0$ (it is a maximum of quantities that are nonnegative). If this maximum is actually equal to 0, it means that $\|Ax\|_2 = 0$ for all x. Since $\|\cdot\|_2$ is a norm, that means $Ax = 0$ for all x. It follows from here that $A = 0$, since we can substitute $x = e_i$ for example to show that the i^{th} column of A is zero.
- Proving that $\|\alpha A\| = |\alpha|\,\|A\|$:
  Proof: by using the fact that $\|\cdot\|_2$ is a norm, the definition of $\|A\|$ implies that

$$\|\alpha A\| = \max_{x\neq 0}\frac{\|\alpha Ax\|_2}{\|x\|_2} = \max_{x\neq 0}\frac{|\alpha|\,\|Ax\|_2}{\|x\|_2} = |\alpha| \max_{x\neq 0}\frac{\|Ax\|_2}{\|x\|_2} = |\alpha|\,\|A\|$$

- Proving that $\|A+B\| \leq \|A\| + \|B\|$:
  Proof: by using the fact that $\|\cdot\|_2$ is a norm, the definition of $\|A\|$ implies that

$$\begin{aligned}
\|A+B\| &= \max_{x\neq 0}\frac{\|Ax+Bx\|_2}{\|x\|_2} \\
&\leq \max_{x\neq 0}\frac{\|Ax\|_2 + \|Bx\|_2}{\|x\|_2} \\
&= \max_{x\neq 0}\left(\frac{\|Ax\|_2}{\|x\|_2} + \frac{\|Bx\|_2}{\|x\|_2}\right) \\
&\leq \max_{x\neq 0}\frac{\|Ax\|_2}{\|x\|_2} + \max_{x\neq 0}\frac{\|Bx\|_2}{\|x\|_2} \\
&= \|A\| + \|B\|
\end{aligned}$$

  The second last step follows from the fact that by allowing the x that maximizes the first term to be different from the x that maximizes the second term, we end up with a quantity that is possibly larger.

2. Image of a circle. Plot the image of a unit circle in $\mathbb{R}^2$ when each point is multiplied by $A = \left(\begin{smallmatrix} 3 & -2 \\ -1 & 5 \end{smallmatrix}\right)$. Also overlay the scaled left singular vectors $\sigma_1 u_1$ and $\sigma_2 u_2$ on your plot and verify that they line up with the axes of the ellipse.

SOLUTION: See the course website for the script `linear_sensitivity.m`, which is the demo we covered in lecture.

3. Dimension reduction. Load the file `sdata.csv` which contains a 1000×3 matrix of data. Each row of the matrix is a point (x_i, y_i, z_i) in $\mathbb{R}^3$. We will approximate this data set as an affine one-dimensional space (a line that doesn't pass through the origin).

a) Find the line that best approximates the data in the sense of minimizing the sum of the squares of the projections of all points onto the line. Plot the line and the data on the same axes and verify that the line approximates the points. *Hint:* before finding the line, shift every point so that the data has zero mean. You can make 3D scatter plots in Matlab by using `plot3`.

b) Instead of using three numbers (x_i, y_i, z_i) to describe each data point, we can now use a single number w_i, which is the position along the line of the projected data point. Give a formula that converts (x, y, z) to w and the reverse formula, which converts w to a point (x, y, z).

c) Convert the data set to w_i coordinates, and plot a histogram of the $\{w_i\}$ to see how the points are distributed. Use 20 equally spaced bins for the histogram.

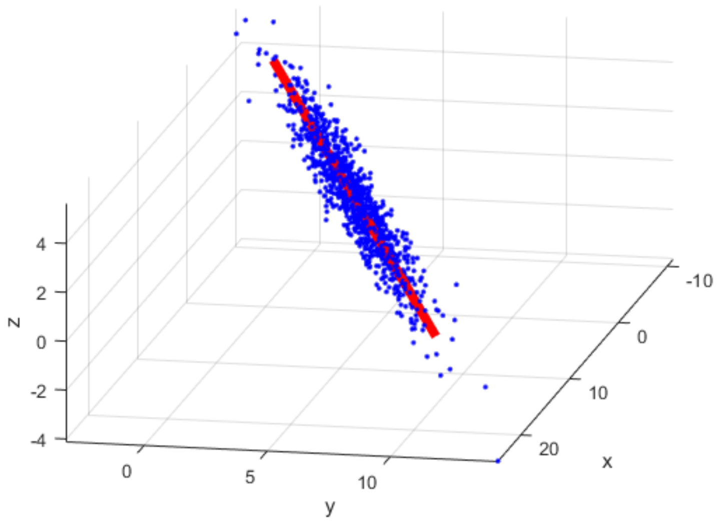
SOLUTION:

a) This is a PCA (SVD) problem similar to the one we did in class as an example. This script reads in the file and makes the plot. We can see that the line approximates the data well.

```
A = csvread('sdata.csv')
xm = mean(A,1);                        % compute mean of all the points
[u,s,v] = svd(A - repmat(xm,N,1));     % subtract mean from each row
v1 = v(:,1);                           % leading right singular vector
s1 = s(1,1);                           % largest singular value

figure(1)
plot3(A(:,1),A(:,2),A(:,3),'b.')
xlabel('x'); ylabel('y'); zlabel('z')
axis equal; grid on

% plot vector v1 (shifted by the mean) together with the data
t = linspace(-1,1);
hold on; plot3( xm(1)+t*s1*v1(1), xm(2)+t*s1*v1(2), ...
                xm(3)+t*s1*v1(3), 'r-','LineWidth',5 )
```



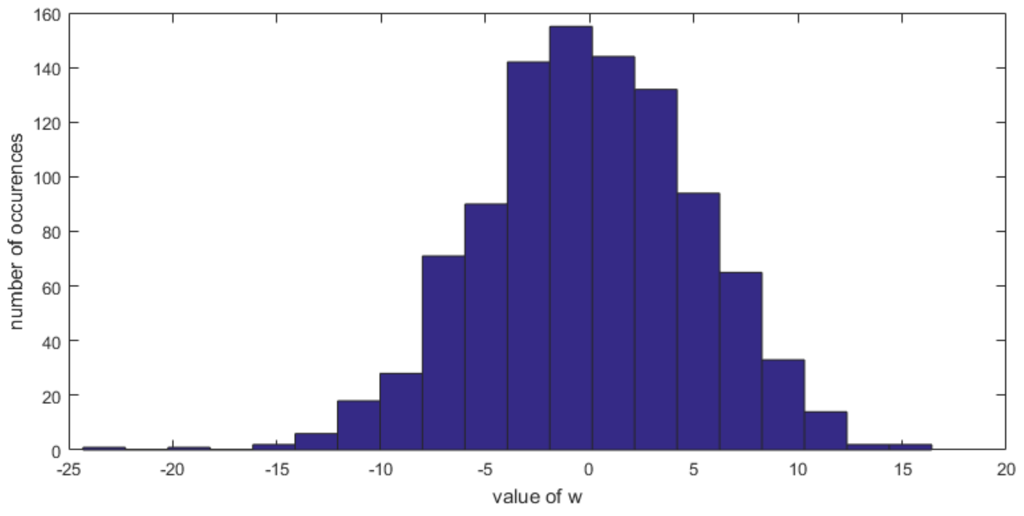
b) The idea here is that we're projecting the points onto the vector v_1, which is the leading right singular vector we plotted in part (a). If $p = (x, y, z)$ is the point in question and $p_m = (x_m, y_m, z_m)$ is the mean value we found in our code, we can convert to position along the line of the projected data by shifting the point to zero, and then taking the inner product with the direction:

$$w = v_1^\mathsf{T}(p - p_m)$$

For the reverse direction, it isn't possible to reconstruct the exact point p just from w, so what we do instead is give the point closest to p along the line described by v_1. This is simply wv_1. Therefore:

$$\widehat{p} = v_1 w + p_m$$

c) Once all points have been shifted and converted using the formulas from part (b), one way to plot the histogram is using the short matlab code: `hist(w,20)`. Here is the result:



4. Image compression. In this problem, we'll use low-rank approximations to compress an image.

a) Load the file `bucky.csv` which contains a matrix $A \in \mathbb{R}^{600\times400}$ of grayscale values scaled to lie between 0 and 1. Plot the image. In Matlab, you can do this via the commands:

```
A = csvread('bucky.csv');
figure; imagesc(A,[0 1])
colormap gray; axis image; axis off
```

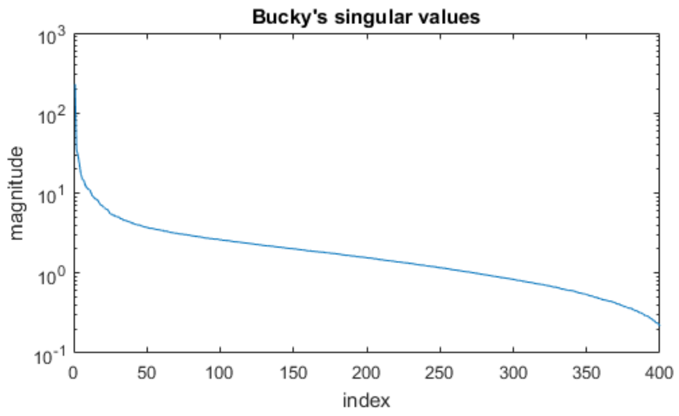
b) Plot the singular values of A. What do you notice?

c) Approximate A as a rank r matrix by only keeping the r largest singular values and making the rest zero. Try this for $r \in \{10, 20, 50, 100\}$ and plot the corresponding compressed images.

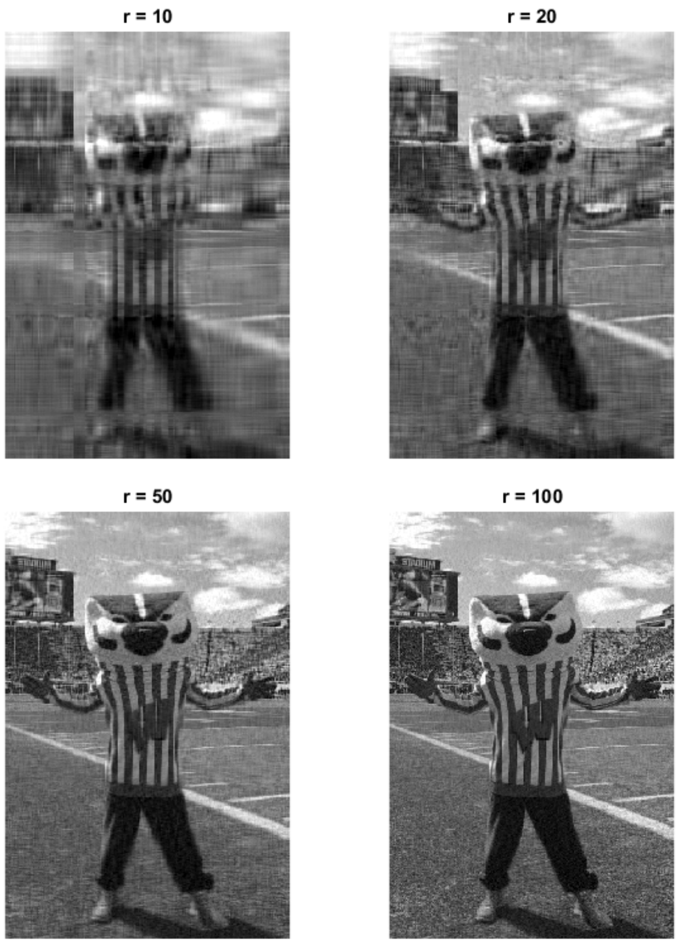
d) Compare the space required to store the full A matrix with the space required to store the rank r approximation of A; how many times smaller is the storage requirement for $r \in \{10, 20, 50, 100\}$? You may assume that storage space requirements are proportional to the number of numbers that must be stored. e.g. a 10×10 matrix contains 100 numbers.

SOLUTION:

a), b) To plot the image, use the code provided. To plot the singular values, we can write: `[u,s,v] = svd(A); semilogy(diag(s));`. I plotted them on a log-y axis to better discern the smaller singular values. Here are the plots:



c) The rank r approximation is `Ar = u(:,1:r)*s(1:r,1:r)*v(:,1:r)'`, so we can adapt the code above to produce the corresponding approximate images. Here they are:



d) The A matrix is 600×400, so storing it requires storing 240,000 numbers. If we use a rank r approximation, we only need to store a factorization, e.g. $U_1\Sigma$ and V_1. These matrices are $600 \times r$ and $400 \times r$, respectively. The ratio of sizes is:

$$\text{savings factor} = \frac{\text{storing full matrix}}{\text{storing rank } r \text{ approx}} = \frac{240000}{600r + 400r} = \frac{240}{r}$$

So with $r = 10, 20, 50, 100$, we have a savings factor of $24, 12, 4.8, 2.4$ respectively. Note that if $r > 240$, it's no longer efficient to store the factorization; we might as well just store the whole matrix.