# CS 111: Sample Problems for Midterm 2 (Sp.19, Matni)

**DO NOT DISTRIBUTE BEYOND THIS CLASS! (i.e. do not post on the internet)**

See Exam e02 on the course GitHub page for midterm rules and syllabus.

**1.** Consider the matrix

$$A = \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix}.$$

Some but not all of the following are eigenvectors of the matrix $A$ above. Identify each vector that is an eigenvector and write the corresponding eigenvalue.

(a) $(1, 1, 1, 1)^T$

(b) $(0, -2, 1, 1)^T$

(c) $(0, 2, -1, -1)^T$

(d) $(1, -1, 1, -1)^T$

(e) $(0, 0, 0, 0)^T$

(f) $(3, -1, -1, -1)^T$

**2.** Let:

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 10000 \end{pmatrix}.$$

What IEEE 64-bit floating-point number represents $\kappa(A)$? Give your answer as a regular number only.

**3.** What is the 8-hex-digit IEEE Standard **32-bit** (i.e. single precision) representation of each of the following floating point numbers?

(a) 1.5

(b) 4.25

(c) -$\epsilon$ ($\epsilon$ is the 32-bit machine epsilon: note the negative sign)

(d) $+\infty$

**4.** The exam will likely have a problem like problem 5 on homework 5 (the loops in floating-point), except that we will give credit for any answer within 10% or so of correct, and we won't ask you to convert between decimal and hexadecimal.

**5.** Consider the following time-series data:

```
t = np.array(range(10))
y = np.array([10.0, 10.2, 9.7, 9.4, 9.6, 7.0, 6.6, 4.9, 2.2, 1.0])
```

**5a.** Express the problem of fitting a straight line of the form

$$p = x_0 + x_1 d$$

to the data as a linear least squares problem

$$Ax \approx b$$

where $x = (x_0, x_1)^T$ is the vector of coefficients of the line described above. What are A and b?

**5b.** Express the problem of fitting a quadratic polynomial of the form

$$p = x_0 + x_1 d + x_2 d^2$$

to the data as a linear least squares problem

$$Ax \approx b$$

where $x = (x_0, x_1, x_2)^T$ is the vector of coefficients of the polynomial.What are A and b?

**6.** Given:

$$A = \begin{pmatrix} 2 & 1 & -2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad \text{and} \quad \Lambda(A) = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

**6a.** Calculate/find all the eigenvalues of:

$$B = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 0 & 1 \\ -2 & 0 & 0 \end{pmatrix}.$$

**6b.** Write Python code that verifies your answer down to an expected calculation. Assume the following is imported first:

```
import numpy as np
import numpy.linalg as npla
```

**7.** Given the following adjacency matrix E that describes a networked web of documents:

$$E = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

**7a.** Sketch the network described by E, showing circles for nodes and arrowed lines for the links. There will be a distinguishing characteristic to this network that appears from your sketch: what is it?

**7b.** How do we derive the link matrix A from E and what is it?

**8.** Briefly summarize the difference between the degree centrality/importance measure of a node in a network and what measure is actually utilized in the PageRank algorithm. Don't write more than 2-3 sentences for this answer (so, emphasis on "briefly"!)

## Solutions to the Problems

**1.** Per each vector:

(a) Yes, because $x = (1, 1, 1, 1)^T$ means $Ax = (0, 0, 0, 0)^T = \lambda x (\lambda = 0)$

(b) Yes, because $x = (0, -2, 1, 1)^T$ means $Ax = (0, -2, 1, 1)^T = \lambda x (\lambda = 1)$

(c) Yes, because $x = (0, 2, -1, -1)^T$ means $Ax = (0, 2, -1, -1)^T = \lambda x (\lambda = 1)$

(d) **No**, because $x = (1, -1, 1, -1)^T$ means $Ax = (4, -2, 0, -2)^T \neq \lambda x$ (no $\lambda$ exists)

(e) Yes, because $x = (3, -1, -1, -1)^T$ means $Ax = (0, 0, 0, 0)^T = \lambda x (\lambda = 0)$

**2.** $\kappa(A) = \|A\|.\|A^{-1}\|$ = 10000 x 1 = 10000

**3.**

(a) 1.5 means sign field (1 bit) = 0, exponent field (8 bits) = 01111111, mantissa (23 bits) = 1000...0, so that makes 0.5 = 0x3FC00000

Similarly,
(b) 4.25 = 0x40880000

(c) The smallest negative, non-zero, number in the machine = 0x80800000 (sign bit is 1, exponent field value is 1, mantissa field = 0).

(d) inf = 0x7f800000 (sign bit is 0, exponent field value is 11111111, mantissa field = 0).

**5a.** 10 data points, means t will be integers 0 thru 9. So, A = [1, t vector] and b = [y vector]:

$$A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \\ 1 & 6 \\ 1 & 7 \\ 1 & 8 \\ 1 & 9 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 10.0 \\ 10.2 \\ 9.7 \\ 9.4 \\ 9.6 \\ 7.0 \\ 6.6 \\ 4.9 \\ 2.2 \\ 1.0 \end{pmatrix}.$$

**5b.** A = [1, t vector, $t^2$ vector] and b = [y vector]:

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \\ 1 & 6 & 36 \\ 1 & 7 & 49 \\ 1 & 8 & 64 \\ 1 & 9 & 81 \end{pmatrix} \quad \text{and b = same as above.}$$

**6a.** Notice that B = $A^T$, so the eigenvalues of B are the same as A. Per $\lambda$(A), these are: -1, 2, and 1.

**6b.**

```
import numpy as np
import numpy.linalg as npla

A = np.array([[2, 1, -2], [1, 0, 0], [0, 1, 0]])
B = A.T
dA, VA = npla.eig(A)
dB, vB = npla.eig(B)

print("Difference in eigenvalues is:", npla.norm(dA - dB))
```

It's also acceptable to have something like this:

```
A = np.array([[2, 1, -2], [1, 0, 0], [0, 1, 0]])
B = A.T
dB, vB = npla.eig(B)
dA = np.array([-1,2,1]) # meh...

print("Difference in eigenvalues is:", npla.norm(dA - dB))
```

**7a.** Once you sketch this network, you'll realize that each link in it is reciprocated (i.e. there are no one-way links in this network).

**7b.** The link matrix, A is a column stochastic type of matrix and is calculated by dividing E by its in-degree vector, that is (in Python):

```
A = E / np.sum(E, 1)
```

Therefore:

$$A = \begin{pmatrix} 0 & 0.5 & 0.5 & 0 & 0.25 \\ 0.33 & 0 & 0 & 0 & 0.25 \\ 0.33 & 0 & 0 & 0 & 0.25 \\ 0 & 0 & 0 & 0 & 0.25 \\ 0.33 & 0.5 & 0.5 & 1 & 0 \end{pmatrix}$$

**8.** Degree (both in- and out-degree) centrality/importance ONLY takes into account how many links a node in a network has. This is a little too simple for an algorithm, like PageRank, which requires "importance" to be measured like in-degree centrality, but also weighs each adjacent node by its own out-degree centrality, so that a link to a high-importance node makes the node itself more important than otherwise.

(optionally, you can *additionally* write and explain the PageRank "formula" here):

$$x_k = \sum_{j \in L_k} x_j / n_j$$

Where $x_k$ is the importance of page $k$, $L_k$ is the set of all pages $j$ with a link to page $k$, $x_j$ is the importance of page $j$, and $n_j$ is the out-degree of page $j$.