Homework 5 Solutions

CAS CS 132: Geometric Algorithms

Due: **Thursday October 19, 2023 at 11:59PM**

Submission Instructions

- Make the answer in your solution to each problem abundantly clear (e.g., put a box around your answer or used a colored font if there is a lot of text which is not part of the answer).
- Choose the correct pages corresponding to each problem in Gradescope. Note that Gradescope registers your submission as soon as you submit it, so you don't need to rush to choose corresponding pages. **For multipart questions, please make sure each part is accounted for.**

Graders have license to dock points if either of the above instructions are not properly followed.

Note. Solutions written here may be lengthy because they are expository, and may not reflect that amount of detail that you were expected to write in your own solutions.

Practice Problems

The following list of problems comes from *Linear Algebra and its Application 5th Ed* by David C. Lay, Steven R. Lay, and Judi J. McDonald. They may be useful for solidifying your understanding of the material and for studying in general. **They are optional, so please don't submit anything for them**.

- 2.1.2, 2.1.4, 2.1.10, 2.1.22, 2.1.27, 2.1.34
- 2.2.1, 2.2.7, 2.3.31

1 Matrix Multiplications

Compute the following matrix multiplications.

A. (3 points)

$$\begin{bmatrix} 1 & -1 & 4 \\ 3 & 0 & 1 \\ 0 & 1 & 1 \\ -3 & 1 & 2 \end{bmatrix} \begin{bmatrix} 2 & 0 & 1 & 1 \\ 0 & 0 & 2 & 3 \\ 1 & 1 & 3 & 5 \end{bmatrix}$$

B. (3 points)

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ -1 \end{bmatrix} \begin{bmatrix} 2 & -1 & 2 & 2 \end{bmatrix}$$

C. (4 points)

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}^T \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

D. (4 points)

$$\begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 & \mathbf{a}_4 & \mathbf{a}_5 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

where $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$, $\mathbf{a}_4$, $\mathbf{a}_5$ are vectors in $\mathbb{R}^n$.

Solution.

A.

$$\begin{bmatrix} 1 & -1 & 4 \\ 3 & 0 & 1 \\ 0 & 1 & 1 \\ -3 & 1 & 2 \end{bmatrix} \begin{bmatrix} 2 & 0 & 1 & 1 \\ 0 & 0 & 2 & 3 \\ 1 & 1 & 3 & 5 \end{bmatrix} = \begin{bmatrix} 6 & 4 & 11 & 18 \\ 7 & 1 & 6 & 8 \\ 1 & 1 & 5 & 8 \\ -4 & 2 & 5 & 10 \end{bmatrix}$$

B.

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ -1 \end{bmatrix} \begin{bmatrix} 2 & -1 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 2 & -1 & 2 & 2 \\ 4 & -2 & 4 & 4 \\ 6 & -3 & 6 & 6 \\ -2 & 1 & -2 & -2 \end{bmatrix}$$

C.

$$\begin{bmatrix}1\\1\\1\\1\\1\\1\end{bmatrix}^T \begin{bmatrix}1&0&1&0&1&0\\0&1&0&1&0&1\\1&0&1&0&1&0\\0&1&0&1&0&1\\1&0&1&0&1&0\\0&1&0&1&0&1\end{bmatrix} \begin{bmatrix}1\\1\\1\\1\\1\\1\end{bmatrix} = \begin{bmatrix}1\\1\\1\\1\\1\\1\end{bmatrix}^T \begin{bmatrix}3\\3\\3\\3\\3\\3\end{bmatrix} = 18$$

D.

$$\begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 & \mathbf{a}_4 & \mathbf{a}_5\end{bmatrix} \begin{bmatrix}0&0&1&0&0\\0&1&0&0&0\\0&0&0&1&0\\1&0&0&0&0\\0&0&0&0&1\end{bmatrix} = \begin{bmatrix}\mathbf{a}_4 & \mathbf{a}_2 & \mathbf{a}_1 & \mathbf{a}_3 & \mathbf{a}_5\end{bmatrix}$$

2 One-to-One and Onto Transformations

For each matrix, determine the corresponding matrix transformation is (1) one-to-one but not onto, (2) onto but not one-to-one, (3) both, or (4) neither. Justify your answer in each part.

A. (3 points) $\begin{bmatrix} 1 & 2 & 1 & 3 \\ 1 & 3 & 1 & 3 \\ 1 & 5 & 1 & 4 \end{bmatrix}$

B. (3 points) $\begin{bmatrix} 5 & 0 & 5 \\ 0 & 0 & 0 \\ 4 & 0 & 7 \end{bmatrix}$

C. (3 points) $\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 3 \\ 0 & 0 & 3 \\ 0 & 0 & 3 \end{bmatrix}$

D. (3 points) $\begin{bmatrix} 1 & -2 & 3 & -4 \\ 0 & 5 & -6 & 7 \\ 0 & 0 & -8 & 9 \\ 0 & 0 & 0 & -10 \end{bmatrix}$

Solution.

A. The transformation cannot be one-to-one since there are more columns than rows, and so its columns are not linearly independent. In particular, the first and third columns are identical, so it is easy to come up with inputs to the transformation that output the same value. The transformation is onto. This can either be shown by reducing the matrix to echelon form and counting the pivots, or by recognizing that it is possible to construct the standard basis vectors as linear combinations of the columns. This is case (2).

B. The transformation is not one-to-one because the columns of the matrix are linearly dependent. This is easy to see since $\mathbf{0}$ is a column. It is also not onto. Since there is an all zeros row, the echelon form will not have a pivot in every row. This is case (4).

C. The transformation is one-to-one. This is maybe easiest to see by reducing it to echelon form and noting that there is a pivot in every column. It is not onto, there are more rows than columns; three vectors cannot span all of $\mathbb{R}^4$. This is case (1).

D. The transformation is both one-to-one and onto. It is already in echelon form and there is a pivot in every column and every row. Its reduced echelon form is the identity matrix. This is case (3).

3 Matrix Inverses

For each of the following matrices, compute its inverse.

A. (3 points) $\begin{bmatrix} 1 & 3 \\ -3 & -2 \end{bmatrix}$

B. (4 points) $\begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 3 & 0 & 0 \end{bmatrix}$

C. (5 points) the matrix implementing the transformation

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \mapsto \begin{bmatrix} x_1 + x_2 \\ -2x_1 - x_3 \\ x_1 + x_3 \end{bmatrix}$$

Solution.

A.

$$\begin{aligned} \frac{1}{1(-2) - 3(-3)} \begin{bmatrix} -2 & -3 \\ 3 & 1 \end{bmatrix} &= \frac{1}{7} \begin{bmatrix} -2 & -3 \\ 3 & 1 \end{bmatrix} \\ &= \begin{bmatrix} (-2/7) & (-3/7) \\ (3/7) & (1/7) \end{bmatrix} \end{aligned}$$

B.

$$\begin{bmatrix} 0 & 0 & (1/3) \\ 0 & (1/2) & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

C.

$$T(\mathbf{e}_1) = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \quad T(\mathbf{e}_2) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad T(\mathbf{e}_3) = \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$$

so T is implemented by

$$\begin{bmatrix} 1 & 1 & 0 \\ -2 & 0 & -1 \\ 1 & 0 & 1 \end{bmatrix}$$

Row reducing the matrix

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ -2 & 0 & -1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

yields the matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 & -1 & -1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 2 \end{bmatrix}$$

so the matrix implementing T^{-1} is

$$\begin{bmatrix} 0 & -1 & -1 \\ 1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

4 Matrix Powers

Compute the following matrix powers. Your solutions must be reduced as much as possible, and must be exact.

A. (4 point) $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}^{2023}$

B. (4 points) $\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}^{2023} \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}^{2024}$

C. (5 points) $\begin{bmatrix} \cos 2 & -\sin 2 & 0 \\ \sin 2 & \cos 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{2023}$

Hint. Remember that multiplying matrices composes their corresponding matrix transformations.

Solution.

A. Checking a couple numbers, we see that the top right value is the only one that changes, it increases by 1 each time.

$$\begin{bmatrix} 1 & 2023 \\ 0 & 1 \end{bmatrix}$$

B. The affect of the exponent on these matrices is that the exponent is applied to each element on the diagonal:

$$\begin{bmatrix} 2^{2023} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} (0.5)^{2024} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Then multiplying these together we get:

$$\begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

C. We have to recognize here that this matrix just rotates vectors about the origin by 2 radians. If we compose this transformation with itself, we have to repeatedly rotate by that amount:

$$\begin{bmatrix} \cos 4046 & -\sin 4046 & 0 \\ \sin 4046 & \cos 4046 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5 Composing Rotations

Consider the following $\mathbb{R}^3$ rotation matrices.

$$A = \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ & 0 \\ \sin 45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos 180^\circ & -\sin 180^\circ \\ 0 & \sin 180^\circ & \cos 180^\circ \end{bmatrix}$$

The matrix A rotates vectors around the x_3-axis by 45 degrees, and B rotates vectors around the x_1-axis by 180 degrees. Also remember that

$$\begin{aligned} \cos 45^\circ &= \frac{\sqrt{2}}{2} \\ \sin 45^\circ &= \frac{\sqrt{2}}{2} \\ \cos 180^\circ &= -1 \\ \sin 180^\circ &= 0 \\ \cos(-\theta) &= \cos(\theta) \text{ for any } \theta \\ \sin(-\theta) &= -\sin(\theta) \text{ for any } \theta \end{aligned}$$

A. (4 points) Determine A^{-1}. Every entry should be a scalar multiple of 1, $\cos 45^\circ$ or $\sin 45^\circ$. In particular, don't include any trigonometric functions applied to negative angles. *Hint.* You don't have to do any calculations for this. Think about what it means to "undo" a rotation.

B. (6 points) Calculate ABA^{-1}. Your solutions should be as reduced as possible, and should not contain any trigonometric functions.

C. (4 points) Describe what the transformation implemented by ABA^{-1} does geometrically. Your answer should be of the form "rotation by [NUMBER] degrees around the span of [VECTOR]." *Hint.* We've seen this transformation before.

Solution.

A. Rotation by -45°:

$$\begin{bmatrix} \cos 45^\circ & \sin 45^\circ & 0 \\ -\sin 45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

B. I'll leave the calculations to you, but you should get

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

C. This is rotation by 180 degrees around the span of $[1\ 1\ 0]^T$.

6 Mini Hill Cipher (Programming)

(15 points) You will be filling in an implementation of a simplified version of a classical cipher called the *Hill Cipher*.

Generally speaking, a cipher is a process for encrypting information; if you want to send information to someone in a secure way, you may want to transform it so that it is difficult for an eavesdropper to interpret, but easy for the receiver to *decode* into the intended message.

The cipher we will be working with is based on a classical cipher invented by Lester Hill in 1929, and uses properties of matrix inverses. The sender and receiver share an invertible matrix A called a *key*. In the case of this assignment, it is a 4×4 matrix. We will be encrypting strings—which we will call *messages*—made up entirely of capital English letters (in particular, no spaces or punctuation). The encryption of a message m goes as follows:

- Convert the string m into a list of numbers between 0 and 25, inclusive, where A goes to 0, B goes to 1, C goes to 2, and so on. So the message ABBAB goes to [0, 1, 1, 0, 1]. This is implemented for you by `message_to_nums`.
- Add 26s to the end of this list of numbers until its length is a multiple of 4. So the list [0, 0, 1, 0, 1] becomes [0, 0, 1, 0, 1, 26, 26, 26]. This is called *padding* and is implemented for you in the function `pad`.
- Convert this list of numbers into a matrix M so that every 4 numbers is a **column from top to bottom** in M. So the list [0, 0, 1, 0, 1, 26, 26, 26] becomes the matrix

  $$\begin{bmatrix} 0 & 1 \\ 0 & 26 \\ 1 & 26 \\ 0 & 26 \end{bmatrix}$$

  You need to implement this in the function `nums_to_matrix`. You should assume that the length of the list is a multiple of 4.

  There is a long and a short way to do this. For the short way, note that

  - `a.T` will give you the transpose of `a`
  - by changing the value of `a.shape`, a 1D array can be made into a 2D array of a particular shape.

  Experiment with these. The long way is to solve this like a standard problem on python lists and then build a numpy array out of your solution.

  Once you've completed this, the function `message_to_matrix`, which combines everything so far into a single step, is implemented for you.

- Encrypt the information in M as AM, the product of A and M. Remember that matrix multiplication in numpy is `A @ M`.

- Convert this new matrix AM back into a string of numbers, and then into a message over a new collection of symbols. This is implemented for you in `matrix_to_code`.

Decoding an encrypted message c is exactly the same process, but in reverse. In particular, rather than multiplying by A, we multiply by A^{-1}.

- Convert c a list of numbers, and then into a matrix C in the same way as above. This is implemented for you in `code_to_matrix`.
- Decrypt the information as $A^{-1}C$, the product of A^{-1} and C. Remember that matrix inversion in numpy is `np.linalg.inv(A)`.
- Convert this new matrix $A^{-1}C$ back into a string of numbers, and then into the original message by flattening the matrix and unpadding it. This is implemented for you in `matrix_to_message`.

This all works because if M is the matrix representing a message and $C = AM$ is the matrix representing the encrypted message, then $A^{-1}C = A^{-1}(AM) = IM = M$, the matrix of the original message.

You are given starter code in the file `hw05prog.py`. **Don't change the name of this file when you submit.** Also don't change the names of the functions included in the starter code. **The only changes you should make are to fill in the TODO items in the starter code.** All together you have to fill in three functions:

- `nums_to_matrix` which converts a list of numbers, whose length is $4k$ into a $4 \times k$ matrix as described above
- `encode`, which implements the above encoding procedure
- `decode`, which implements the above decoding procedure

You will upload a single file `hw05prog.py` to Gradescope with your implementations of the required functions. You may use any functionality from numpy for your implementation. Please read the code carefully, much of this assignment is understanding what the given code does.

Note. This is not a secure cipher. Because it's based on linear algebra, the techniques that we've been developing in this course can be used figure out the key matrix, often from a single message-code pair. It's also worth noting that the actual implementation of the Hill cipher is much more general and uses modular arithmetic. If you're interested, I highly recommend looking into it. It is a good weekend project to implement the actual Hill cipher in your favorite programming language.