CS545 Problem Set 1

Problem 1.a

Assume a, b are two arbitrary column vectors with size n, m such that: $a = [a_1, a_2, a_3, \cdots, a_n]^\top$, $b = [b_1, b_2, b_3, \cdots, b_m]^\top$, then, $a \cdot b^\top$ will be equal to:

$$a \cdot b^\top = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} \cdot [b_1 \cdots b_m] = \begin{bmatrix} a_1b_1 & a_1b_2 & a_1b_3 & \cdots & a_1b_m \\ a_2b_1 & a_2b_2 & a_2b_3 & \cdots & a_2b_m \\ \vdots & & & & \\ a_nb_1 & a_nb_2 & a_nb_3 & \cdots & a_nb_m \end{bmatrix}$$

vectorizing $a \cdot b^\top$, and we can find its expanded equation is equivalent to the expanded form of $b \otimes a$:

$$\text{vec}(a \cdot b^\top) = \text{vec}(\begin{bmatrix} a_1b_1 & a_1b_2 & a_1b_3 & \cdots & a_1b_m \\ a_2b_1 & a_2b_2 & a_2b_3 & \cdots & a_2b_m \\ \vdots & & & & \\ a_nb_1 & a_nb_2 & a_nb_3 & \cdots & a_nb_m \end{bmatrix}) = \begin{bmatrix} a_1b_1 \\ a_2b_1 \\ \vdots \\ a_nb_1 \\ \hline a_1b_2 \\ a_2b_2 \\ \vdots \\ a_nb_2 \\ \hline a_1b_m \\ a_2b_m \\ \vdots \\ a_nb_m \end{bmatrix} = \begin{bmatrix} b_1 \cdot a \\ b_2 \cdot a \\ b_3 \cdot a \\ \vdots \\ b_m \cdot a \end{bmatrix} = b \otimes a$$

Problem 1.b

Assume we have two arbitrary matrices A, B with same order, assume each of them have dimension $m \times n$, then, we can represent the trance of $A^\top B$ as:

$$\begin{aligned} \text{tr}(A^\top \cdot B) &= \sum_{k=1}^{n} (A^\top B)_{kk} \\ &= \sum_{k=1}^{n} A_{\cdot k}^\top B_{\cdot k} \\ &= [A_{\cdot 1} \quad A_{\cdot 2} \quad A_{\cdot 3} \quad \cdots \quad A_{\cdot n}] \begin{bmatrix} B_{\cdot 1} \\ B_{\cdot 2} \\ B_{\cdot 3} \\ \vdots \\ B_{\cdot n} \end{bmatrix} \\ &= \text{vec}(A)^\top \cdot \text{vec}(B) \end{aligned}$$

Problem 1.c

According to the definition, the matrix A will be semi-definite if and only if the real number $z^T Az$ is non-negative for every nonzero real column vector z.

Now, assume that $A = X \cdot X^\top$ where X is a real matrix, and z is an arbitrary vector with dimension equal to the row number of A. Then:

$$z^\top Az = z^\top (X \cdot X^\top) z$$

by matrix multiplication property, we can change the grouping surrounding matrix multiplication, that is:

$$z^\top (X \cdot X^\top) z = (z^\top X) \cdot (X^\top z)$$

assume a matrix $u = z^\top X$, by the property of transpose that the product of the transposes of two matrices in reverse order is equal to the transpose of the product of them, that is $u^\top = X^\top z$, then:

$$(z^\top X) \cdot (X^\top z) = u \cdot u^T = u_1^2 + u_2^2 + u_3^3 + \cdots u_n^2 = ||u||^2$$

which is equal to the square of the L2 norm for matrix u, and it is a non-negative number. Therefore, we can say:

$$z^T A z \in \mathbb{R}_+$$

that is $X \cdot X^T$ is semi-definite.

Problem 2

let's define the event of the woman has breast cancer is B, and get positive mammogram to be M. From the information given, we know:

$$P(B) = 0.6\%$$
$$P(M|B) = 90\%$$
$$P(M|\neg B) = 7\%$$

what we need to calculated is $P(B|M)$, applied the Bayes' theorem,

$$P(B|M) = \frac{P(M|B)P(B)}{P(M)}$$

we can notice that except $P(M)$, the rest terms are all given, so we need to solve for $P(M)$ firstly, and this term represent the probability of woman get positive mammogram.

$$\begin{aligned} P(M) &= P(M|B)P(B) + P(M|\neg B)P(\neg B) \\ &= 0.9 \cdot 0.006 + 0.07 \cdot (1 - 0.006) \\ &= 0.07498 \end{aligned}$$

then, plug-in all term into the Bayes' formula:

$$P(B|M) = \frac{0.9 \cdot 0.006}{0.07498} \approx 7.2\%$$

Problem 3.1.a

assume the input matrix is $G_{MN \times K}$, define a column vector $K_l = [\underbrace{\frac{1}{K}; \frac{1}{K}; \cdots; \frac{1}{K}}_{K}]$, then the mean image can be expressed as:

$$\text{Mean Image} = (G \cdot k_l)^{(M)}$$

Problem 3.1.b

assume the input matrix is $G_{MN \times K}$, define a matrix M_s with shape $2 \times 2N$ and equal to:

$$M_s = \begin{bmatrix} 1 & 0 & 1 & 0 & \cdots \\ 0 & 1 & 0 & 1 & \cdots \end{bmatrix}$$

then, the mean value for top and bottom half for the K images, let's called it $D_{2 \times K}$, can be calculated by:

$$D = M_s \otimes J_{1, M/2} \cdot \frac{1}{MN/2} \cdot G$$

where the $J_{1,M/2}$ represent the row vector of ones with length $M/2$. We also need to define the mean vector of D to be μ, it can be obtained by:

$$\mu = D \cdot J_{K,2} \cdot \frac{1}{K}$$

the calculation above resulting μ to be a column vector with length 2, finally we can express the covariance matrix:

$$\text{Cov} = \frac{(D-\mu)(D-\mu)^\top}{K-1}$$

Problem 3.2.a

assume the input matrix is $T_{M,N,3,K}$, we first operate both $\text{vec}(\cdot)$ and vec-transpose on it to reduce its dimension to be 2.

$$(\text{vec}(T))^{(3MN)}$$

after this this step, the input matrix will be a 2-dim matrix with size $3MN \times K$, noticed that each row represent the same color, so we can easily find the mean vector of it with respect to its row. Let's call the resulting vector A_1:

$$A_1 = (\text{vec}(T))^{(3MN)} \cdot \left.\begin{bmatrix} 1/k \\ 1/k \\ \vdots \\ 1/k \end{bmatrix}\right\}\text{K}$$

the resulting matrix is a column vector with a length of $3MN$, and it exhibits the pattern $R^{MN}G^{MN}B^{MN}$. To align all the R, G, and B pixels into their respective columns, we can utilize the vec-transpose operation. Let's call the resulting matrix A_2:

$$A_2 = A_1^{(MN)}$$

A_2 has the shape $MN \times 3$, the reason this makes sense is that we already know the exact number of R, G, and B pixels is MN. Then, we need to find the mean vector for A_2 wrt its row, then, to transpose the mean vector into the mean image matrix, simply do a vec-transpose:

$$\text{Mean Image} = (A_2 \cdot \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix})^{(M)}$$

Problem 3.2.b

Because we only need the mean of red channel, so we can first use the color mixing formula to keep the read channel and set the other to be zero

$$(\text{diag}\,[1 \quad 0 \quad 0] \otimes I \otimes I)\text{vec}(T)$$

the resulting vector is a column vector with size $3MNK$, we can do a vec-transpose to make each image be in an unique column:

$$T_{red} = ((\text{diag}\,[1 \quad 0 \quad 0] \otimes I \otimes I)\text{vec}(T))^{(3MN)}$$

the matrix T_{red} has the shape $3MN \times K$, for each column vector is an image, and the first MN elements in each column are the values for red channel, and the rest are values for G, B which are already been set as zero in the first step. Then, we need to obtain the mean value of red channel among the K images, and convert it back to matrix by using vec-transpose:

$$M_r = (T_{red} \cdot J_{K,1} \cdot \frac{1}{K})^{(MN)}$$

the $J_K, 1$ represent the column vector of ones with size K, this operation resulting a matrix with three column vectors, each represent the mean value for R, G, B (column for G, B are zeros). Because we only need the red channel, we need to use a filter to remove the last two columns. After removing, do the vec-transpose to convert it back to a matrix:

$$\text{Mean Image}_{red} = (M_r \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix})^{(M)}$$

Problem 4

Given a matrix A that represents our transformation, our objective is to deduce its dimensions and establish the elements within it. Let's designate the resulting vector from our transformation as S, which represents the complex spectrogram coefficients vector. Given an input sound vector of size N, the number of frames can be deduced based on the given DFT_size and Hop_size :

$$\text{Frame\#} = \left\lceil \frac{N - \text{DFT\_size}}{\text{Hop\_size}} \right\rceil + 1 = \left\lceil \frac{N - 64}{32} \right\rceil + 1$$

Each frame encompasses DFT_size frequency bins. These frequency bins then constitute our resulting vector S. The length of S can thus be given as:

$$|S| = \left\lceil \frac{N - 64}{32} \right\rceil \times 64 = 2N - 64$$

Based on the properties of matrix multiplication, this is also the number of rows in matrix A .
Summarizing, matrix A should be of size $(2N - 64) \times N$.

Now, let's discern the value of each entry in this matrix. To compute the complex spectrogram coefficient for a segment $[a_1, a_2, \ldots, a_{64}]$, firstly, a Hann window is applied to mitigate spectral leakage:

$$w(n) = 0.5 \times \left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right), \quad N = 64$$

The resulting data after applying the Hann window becomes:

$$\text{data\_with\_hann\_window} = [w(1)a_1, w(2)a_2, \ldots, w(64)a_{64}]$$

The subsequent step is to apply the DFT:

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{-j\left(\frac{2\pi k n}{N}\right)}$$

Given $x = \text{data\_with\_hann\_window}$ and $N = 64$, the transformed data is:

$$\text{data\_with\_hann\_window\_DFT} = [X(0), X(1), \ldots, X(63)]$$

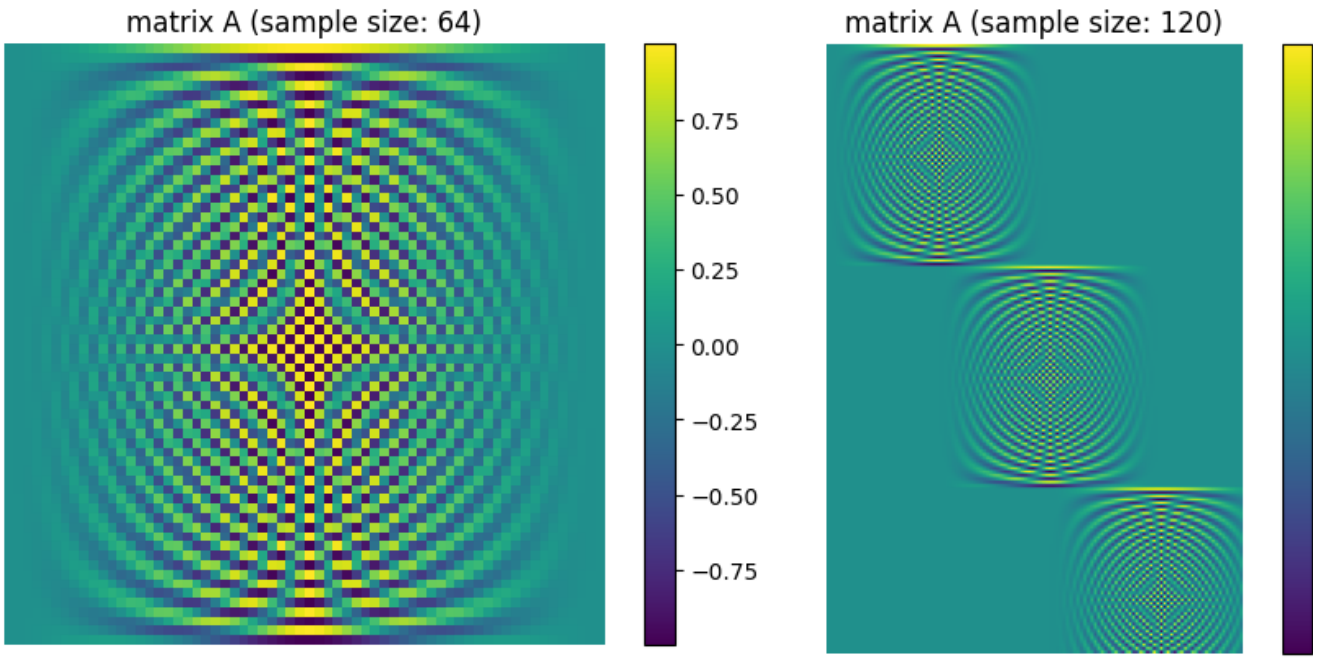
To represent this process in matrix form and considering the hop size, let's define three functions:

$$\begin{aligned} X(n,k) &= e^{-j(2\pi n k/64)} \\ H(k) &= 0.5 \times \left(1 - \cos\left(\frac{2\pi k}{63}\right)\right) \\ q(i) &= 32\left(\left\lfloor \frac{i}{64} \right\rfloor\right) \end{aligned}$$

For matrix A with its element at the i^{th} row and j^{th} column represented as $A_{i,j}$, the entry has to satisfy the following conditions:

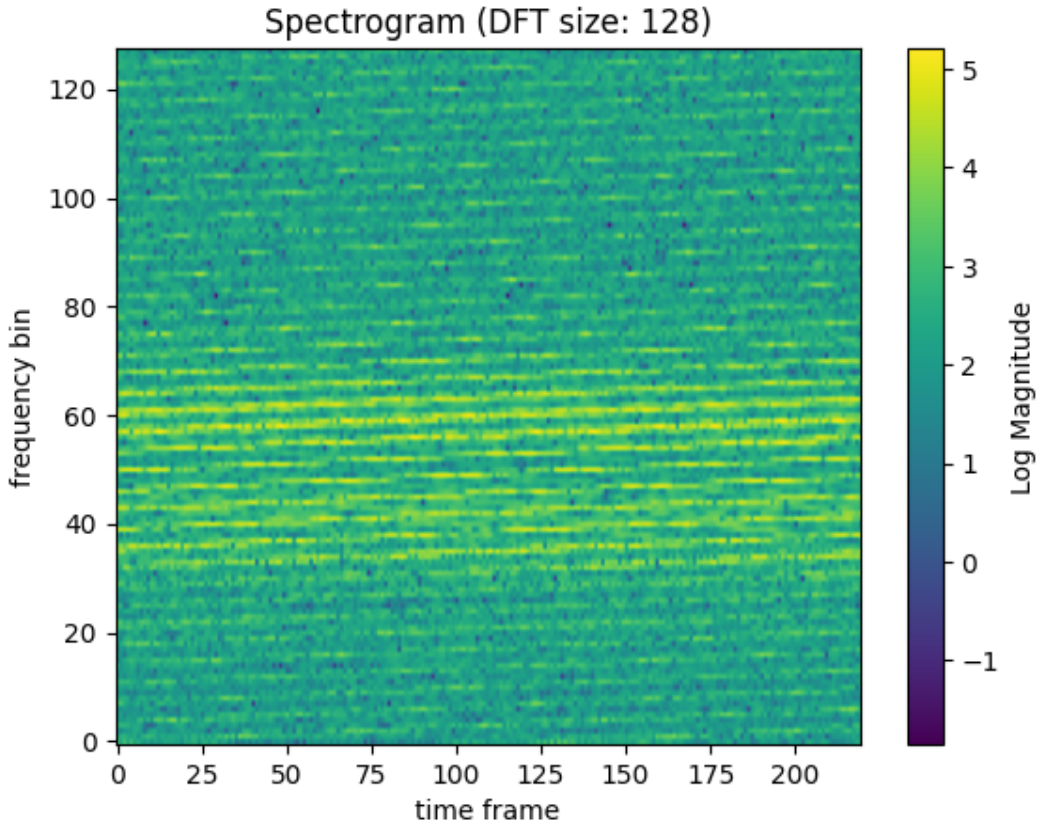
$$A_{i,j} = \begin{cases} X(n = j - q(i), k = i \mod 64) \cdot H(k = j - q(i)) & \text{if } q(i) \le j \le q(i) + 64 \\ 0 & \text{otherwise} \end{cases}$$

To visualize the matrix A, I I employed sample sounds of sizes 64 and 120, utilizing $\mathrm{DFT\_}size = 64$ and $\mathrm{Hop\_}size = 32$



Extra point

To generate a spectrogram via using the column vector S we obtain from the previous question, we can do a vec-transpose with order DFT_{size}, and sum up the real and imaginary part, normalize the amplitude by taken Log_{10}, then draw the heat map for the resulting matrix, I utilized Python for this process, applying it to a recording of my voice. The resultant spectrogram is presented below:"



p.s. I'm uncertain about the appropriate handling of the amplitude. I derived it by summing the real and imaginary parts, which I suspect might not be the optimal approach. Consequently, the resulting amplitude appears excessively large before normalizing

```
import tqdm
import matplotlib.pyplot as plt
import numpy as np
from pydub import AudioSegment

sample_size = 20000
DFT_size = 128
Hop_size = 64

DRAW_A = False
DRAW_SPECTROGRAM = True
SAMPLE_FILENAME = 'sample.m4a'
assert (DFT_size / 2 == Hop_size)


# DFT function
def dft(n, k):
    return np.exp(-1j * (2 * np.pi * n * k / DFT_size))


# Hann window function
def hann(k):
    return 0.5 * (1 - np.cos(2 * np.pi * k / (DFT_size - 1)))


# down sampling, or computer will bomb
sample_raw = AudioSegment.from_file(SAMPLE_FILENAME).set_frame_rate(8000)
sample = np.array(sample_raw.get_array_of_samples())
N = min(sample_size, sample.shape[0])
sample = sample[:min(sample_size, N)]
print(f"sound has shape: {sample.shape[0]}, we use {N} datas as sample")

A = np.zeros((2 * N - DFT_size, N))

for i, row in enumerate(tqdm.tqdm(A)):
    q = Hop_size * (i // DFT_size)
    for j in range(row.shape[0]):
        if (j >= q) and (j <= (q + DFT_size)):
            # A_ij != 0
            row[j] = dft(n=j - q, k=i % DFT_size) * hann(j - q)

        # otherwise A_ij = 0

# visualize A
if DRAW_A:
    plt.imshow(A.real)
    plt.colorbar()
    plt.title(f"matrix A (sample size: {sample_size})")
    plt.axis('off')
    plt.show()

# draw spectrogram
if DRAW_SPECTROGRAM:
    spec_coefficient = A @ sample
    # keep partial of spec coefficient which can be | 64, add up real and imaginary
part, and normalize log10
    spec_coefficient = np.abs(spec_coefficient[:spec_coefficient.shape[0] // DFT_size *
DFT_size])
```

```
    spec_coefficient = spec_coefficient.reshape((DFT_size, -1))

    plt.imshow(np.log10(spec_coefficient), origin='lower', aspect='auto')
    plt.colorbar(label='Log Magnitude')
    plt.title(f"Spectrogram (DFT size: {DFT_size})")
    plt.xlabel('time frame')
    plt.ylabel('frequency bin')
    plt.show()
```