HW8

May 30, 2020

1 CS 168 Spring Assignment 8

SUNet ID(s): 05794739

Name(s): Luis A. Perez

Collaborators: None

By turning in this assignment, I agree by the Stanford honor code and declare that all of this is my own work.

2 Imports

[1]:
```
import collections
import matplotlib.pyplot as plt
import scipy

import numpy as np
from PIL import Image
from sklearn import decomposition
import pandas as pd
import seaborn as sns
import os
import warnings

import IPython

from typing import Dict, List, Text, Tuple

# Make figure larger
plt.rcParams['figure.figsize'] = [10, 5]

# Set numpy seed for consistent results.
np.random.seed(1)
```

[2]:
```
class Globals:
    DATA_PATH = 'data/'
```

3 Part 2

3.1 Part 2c

[144]:
```
from scipy import fft

def convolve(x: List[int], y:List[int]):
    """Compute x*y.

    Only accepts real-valued x,y.
    """
    m, l = len(x), len(y)
    n = max(m, l)
    x = x + [0] * (n + max(n - m, 0))
    y = y + [0] * (n + max(n - l, 0))
    assert len(x) == 2*n
    assert len(y) == 2*n
    return np.rint(np.real(fft.ifft(fft.fft(x) * fft.fft(y))[:m + l - 1])).
↪astype(int)


def multiply(x: List[int], y: List[int]):
    """Multiplies x and y.

    Args:
        x: A list of digits. Lower indeces represent lower digits.
        y: Same as ax.

    Returns:
        A list of the same format representing the product x*y.
    """
    # Convolve the two using FFT.
    product = convolve(x, y)
    # Limit values to just be single digit.
    carry = 0
    fixed_product = []
    for val in product:
        digit = (val + carry) % 10
        carry = (val + carry) // 10
        fixed_product.append(int(np.rint(digit)))

    while carry > 0:
        digit = carry % 10
        carry = carry // 10
        fixed_product.append(int(np.rint(digit)))
    return fixed_product

def to_list(x: str) -> List[int]:
```

```
    return [int(char) for char in reversed(x)]

def from_list(x: List[int]) -> str:
    return "".join([str(y) for y in reversed(x)])
```

[145]:
```
def problem2c():
    x = "12345678901234567890"
    y = "98765432109876543210987654321O"
    print(f"{x} x {y} = {from_list(multiply(to_list(x), to_list(y)))}")
```

[146]:
```
problem2c()
```

```
12345678901234567890 x 98765432109876543210987654321O =
121932631137021795224965706422374638011112635269OO
```

4 Problem 3

[163]:
```
from scipy.io import wavfile
```

[164]:
```
def load_wav():
    with open(os.path.join(Globals.DATA_PATH, 'laurel_yanny.wav'), 'rb') as f:
        sampleRate, data = wavfile.read(f)
    return sampleRate, data
```

[165]:
```
def save_wav(sampleRate, data, outfile='output.wav'):
    data = (data * 1.0 / np.max(np.abs(data))*32767).astype(np.int16)
    with open(os.path.join(Globals.DATA_PATH, outfile), 'wb') as f:
        wavfile.write(f, sampleRate, data)
```

[167]:
```
# Test.
save_wav(*load_wav())
```

4.1 Problem 3b

[168]:
```
def problem3b():
    sampleRate, data = load_wav()
    plt.title('Waveform for Laurel/Yanny Wavefile')
    plt.xlabel('Time (seconds)')
    plt.ylabel('Displacement')
    plt.plot(np.arange(len(data)) / sampleRate, data)
    plt.savefig('figures/laurel_yanny_waveform.png', format='png')
    plt.close()
```

[169]:
```
problem3b()
```

4.2 Problem 3c

[171]:
```
def problem3c():
    sampleRate, data = load_wav()
    data = fft.fft(data)

    plt.title('Fourier Transform of Waveform for Laurel/Yanny Wavefile')
    plt.xlabel('Frequency (Hz)')
    plt.ylabel('Amplitude')
    # Only plot magnitudes of values.
    plt.plot(np.arange(len(data)), np.abs(data))
    plt.savefig('figures/laurel_yanny_waveform_fft.png', format='png')
    plt.close()
```

[172]:
```
problem3c()
```

4.3 Problem 3d

[207]:
```
def problem3d():
    block_size = 500
    max_freq = 80
    sampleRate, data = load_wav()
    heatmap = []
    # Ignore the last chunk.
    for i in range(0, len(data) - block_size, block_size):
        block = data[i:min(i+block_size, len(data))]
        transform = np.abs(fft.fft(block))
        heatmap.append(transform[:max_freq])
    print(f'We have {len(heatmap)} chunks.')
    heatmap = np.stack(tuple(heatmap)).T
    plt.imshow(np.log(heatmap), cmap='hot')
    plt.title(f'Spectogram using Block Size of {block_size}')
    plt.ylabel('Frequency')
    plt.xlabel('Block Index')
    plt.savefig('figures/spectogram.png', format='png')
    plt.close()
```

[208]:
```
x = problem3d()
```

```
We have 86 chunks.
```

4.4 Problem 3d

[222]:
```
def problem3e():
    sampleRate, data = load_wav()

    for t in range(100, 5000, 100):
        transform = fft.fft(data)
```

```
        low_only, high_only = transform.copy(), transform.copy()
        low_mask = (np.arange(len(low_only)) < t) | (np.arange(len(low_only)) >␣
↪(len(low_only) - t))
        low_only[~low_mask] = 0
        high_only[low_mask] = 0
        low_only = fft.ifft(low_only)
        high_only = fft.ifft(high_only)

        save_wav(sampleRate, low_only, outfile=f'[T={t}]low_only.wav')
        save_wav(sampleRate, high_only, outfile=f'[T={t}]high_only.wav')
```

[223]:
```
problem3e()
```

```
/Users/nautilik/.virtualenvs/cs168/lib/python3.7/site-
packages/ipykernel_launcher.py:2: ComplexWarning: Casting complex values to real
discards the imaginary part
```

4.5 Problem 3g

[252]:
```
def problem3g():
    sampleRate, data = load_wav()
    for scale in np.arange(0.25, 2.25, 0.05):
        scaled_sample = int(scale * sampleRate)
        save_wav(scaled_sample, data, outfile=f'[Rate={scaled_sample}]_output.
↪wav')
```

[253]:
```
problem3g()
```

[ ]:
```

```