BLG 354E Signals and Systems for CE (Spring 2018)
Homework-2

Onat Sahin - 150150129
April 5, 2018

Codes for the questions 2,6 and 8 are python 3 codes. Required modules are numpy, scipy, matplotlib, sklearn and skimage

1 Question 1

Continuous time Fourier transform:

$X(jw) = \int_{-\inf}^{\inf} x(t)e^{-jwt}dt$ (CTFT)

We need to convert this to discrete time. For that, we need to get samples from the continuous time version.

T_s = Sampling period

Instead of t, we use $T_s n$. Since we are using samples, we take the sum instead of integration.

$X(e^{jwT_s}) = \sum_{-\inf}^{\inf} x[n]e^{-jwT_s n}, w.T_s = \hat{w}$

$X(e^{j\hat{w}}) = \sum_{-\inf}^{\inf} x[n]e^{-j\hat{w}n}$ (DTFT)

If we evaluate DTFT at a discrete set of equally spaced frequencies $\hat{w_k} = (2\pi/N)k for k = 0, 1, ..., N-1$:

$X(e^{j(2\pi/N)k}) = \sum_{n=0}^{L-1} x[n]e^{-j(2\pi/N)kn} = x[k]$ (DFT)

2 Question 2

My code for this question and voice files are located in the question2 folder.

For the question, I recorded my voice and created my dataset. For the dataset to be somewhat mixed, I ordered the voices like: one Önevoice, one Twovoice, one Önevoice and so on. After this, I followed the steps given in the function and calculated accuracy. With the first 10 coefficients of data and 1 PCA component, my code was able to achieve 100% accuracy. Increasing the number of coefficients and components did not change the outcome. My code:

```python
#Onat Şahin 150150129 Question 2
import numpy as np
from scipy.io import wavfile
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline

ones = []
twos = []

for i in range(1,21):  #Reading the .wav files
    rate1, data1 = wavfile.read("one" + str(i) + ".wav")
    rate2, data2 = wavfile.read("two" + str(i) + ".wav")
    ones.append(np.fft.fft(data1.T[0]))
    twos.append(np.fft.fft(data2.T[0]))

trainFeatures = []
labels = []

for i in range(15): #Arranges the data set so that ones and twos are mixed
    trainFeatures.append(ones[i][:10]) #Taking the first ten coefficients for each data in the set
    trainFeatures.append(twos[i][:10])
    labels.append(1) #Creating the labels for the trainSet
    labels.append(2)

labels = np.array(labels)
trainFeatures = np.matrix(trainFeatures)

tests = ones[15:20] + twos[15:20] #Creating the test set and labels
testLabels = [1,1,1,1,1,2,2,2,2,2]
```

```python
#for i in range(15,20):         #Code to create a mixed test set like the training set. Gives the same result
    #print("ones",ones[i])      #as the set above
    #tests.append(ones[i])
    #print("twos",ones[i])
    #tests.append(twos[i])
    #testLabels.append(1)
    #testLabels.append(2)

for i in range(len(tests)): #Taking the first ten coefficients of the test set
    tests[i] = tests[i][:10]

tests = np.array(tests)

pca = PCA(n_components=1)
logreg = LogisticRegression()
pipeline = Pipeline([('pca', pca), ('logreg', logreg)]) #The pipeline to do the operations
pipeline.fit(trainFeatures, labels) #Training

predictions = []
for test in tests: #Predicting the test set
    predictions.append(pipeline.predict(test))
print("Predictions: ", predictions) #Printing the predictions and true classes
print("True labels: ", testLabels)
correct = 0
for i in range(len(tests)):
    if(predictions[i].item(0) == testLabels[i]):
        correct += 1

print("Accuracy: ", correct/10*100, "%") #Printing the accuracy
```

3 Question 3

Unit impulse is a pulse which starts and ends at the same instant of time because it's duration is very short.

$$\text{Unit impulse} = \delta(t) = \begin{cases} 1 & t = 0 \\ 0 & t \neq 0 \end{cases} \text{ and } \int_{-\inf}^{\inf} \delta(t)dt = 1$$

Unit impulse response is the output of a system when the input is the unit impulse.

4 Question 4

$x[n] = ... + x[-1]\delta[n+1] + x[0]\delta[n] + x[1]\delta[n-1] + x[2]\delta[n-2] + ...$

$x[n] = \sum_{k=-\inf}^{\inf} x[k]\delta[n-k]$

For a n LTI system T:

$y[n] = T\{x[n]\}$

$y[n] = T\{\sum_{k=-\inf}^{\inf} x[k]\delta[n-k]\}$ Because of linearity:

$y[n] = \sum_{k=-\inf}^{\inf} T\{x[k]\delta[n-k]\}$

$y[n] = \sum_{k=-\inf}^{\inf} x[k]T\{\delta[n-k]\}$ Because of time invariance:

$y[n] = \sum_{k=-\inf}^{\inf} x[k]h[n-k] = x[n] * h[n]$ (Convolution) Intuition?

5 Question 5

a) It is not causal because it depends on a future time. It is stable because x and y can be bounded.

b) It is causal because it does not depend on a future value. However, it is not stable. Since integration starts from $-\inf$, it can not be bounded.

c) It is causal because it only depends on the past. It is stable because the result is bounded between 0 and 1.

d) It is causal because it only depends on the present. It is stable because the result is bounded between 0 and 1.

6 Question 6

$y[n] = \{6, 10, 18, 16, 18, 12, 8, 2\}$. The code can be found in q6.py file

7 Question 7

The operation done in the implementation of question 6 can be written as a matrix multiplication operation.

$$y[n] = \begin{bmatrix}2 & 4 & 6 & 4 & 2\end{bmatrix} \begin{bmatrix}3 & -1 & 2 & 1 & 0 & 0 & 0 & 0\\ 0 & 3 & -1 & 2 & 1 & 0 & 0 & 0\\ 0 & 0 & 3 & -1 & 2 & 1 & 0 & 0\\ 0 & 0 & 0 & 3 & -1 & 2 & 1 & 0\\ 0 & 0 & 0 & 0 & 3 & -1 & 2 & 1\end{bmatrix}$$

This multiplication operation gives the same result given to question 6

8 Question 8

convolve2d library function computes the convolution of two 2d signals. In this example, one of the inputs is an image while the other one is a filter. The function basically calculates new pixel values by multiplying the filter with part of an image centered around the calculated pixel. This way, different effects can be achieved depending on the filter. We use a smoothing filter in this example. In the figure below, the first image is the given input image, while the second image is the result of filtering. The colors are different because I had to convert the image to grayscale to be able to represent it as a 2d array. It can be seen that the edges are smoother in the output image.