dsp-assignment-2

February 8, 2020

1 Assigment 2

by Katon D. R. Wibowo (ID: 51934701) Importing required module

[1]:
```
import numpy as np
import pandas as pd
from PIL import Image as Pimage
#from IPython.display import Image
```

1.1 Part 1: Huffman coding

1.1.1 a. Huffman coding function

Numpy was used to create frequency table.Pandas was used to manage array

[4]:
```
def codebook_generator(data):
    #compute a probability table
    freq_table = np.array(np.unique(data, return_counts=True))
    freq_table = np.transpose(freq_table)
    freq_table = pd.DataFrame(freq_table)
    freq_table[1] = pd.to_numeric(freq_table[1])
    freq_table = freq_table.sort_values(by=1, ascending=False)
    #building huffman tree
    codebook = freq_table
    codebook = codebook.reset_index(drop=True)
    codebook[1] = ''
    keys = codebook[1]

    for i in range(0, len(keys)//2):
        b = i*2
        keys.loc[b:b+1] = keys.loc[b:b+1] + '0'
        keys.loc[b+2:] = keys.loc[b+2:] + '1'
        keys.loc[b] = keys.loc[b] + '0'
        keys.loc[b+1] = keys.loc[b+1] + '1'

    codebook[1] = keys
    codebook_list = codebook.values.tolist()
```

```
    codebook = dict(codebook_list)
    return codebook

def huffman_encoding(data, codebook):
    code = ''
    for i in data:
        code = code + codebook[i]
    return code
```

1.1.2 b. Huffman decoding function

[5]:
```
def huffman_decoding(data, codebook):
    # flip codebook
    codebook = {value:key for key, value in codebook.items()}

    # encode data


    return data
```

1.1.3 c. Demonstration

[6]:
```
# sample data
data = [114, 20, 114, 114, 110, 12, 117, 85, 114, 118, 114, 114, 114, 6, 70,␣
 ↪71, 93, 102, 72, 114, 114, 53, 117, 74, 117, 114, 114, 14, 102, 14, 117, 114]
data = np.array(data)

# generating codebook
codebook = codebook_generator(data)

# encoding data
encoded_data = huffman_encoding(data, codebook)

# decoding data
#decoded_data = huffman_decoding(encoded_data, codebook)
```

```
/home/katon/.local/lib/python3.6/site-packages/pandas/core/indexing.py:670:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_with_indexer(indexer, value)
```

1.2 Part 2: Differential and RL coding

1.2.1 a. Differential coding:

1.2.2 encoding

[2]:
```
def diff_encoding(data):
    return code

data = [114, 20, 114, 114, 110, 12, 117, 85, 114, 118, 114, 114, 114, 6, 70,␣
↪71, 93, 102, 72, 114, 114, 53, 117, 74, 117, 114, 114, 14, 102, 14, 117, 114]
data = np.array(data)
data
#new_text = diff_encoding(data)
#new_text
```

[2]:
```
array([114,  20, 114, 114, 110,  12, 117,  85, 114, 118, 114, 114, 114,
         6,  70,  71,  93, 102,  72, 114, 114,  53, 117,  74, 117, 114,
       114,  14, 102,  14, 117, 114])
```

1.2.3 decoding

[8]:
```
def diff_decoding(code):
    return data
```

1.2.4 b. Run-length coding:

1.2.5 encoding

[9]:
```
def rl_encoding(data):
    return code
```

1.2.6 decoding

[10]:
```
def rl_decoding(code):
    return data
```

1.2.7 c. Demonstration

1.2.8 Differential encoding + Huffman encoding

[11]:
```
# differential encoding
data_211 = diff_encoding(data)

# huffman encoding
codebook_21 = codebook_generator(data_211)
data_212 = huffman_encoding(data_211, codebook_21)
```

```
  File "<ipython-input-11-74ec1a74fbb7>", line 2
= huffman_encoding()
^
IndentationError: unexpected indent
```

1.2.9 RLE + Huffman encoding

[ ]:
```
# run-length encoding
data_221 = rl_encoding(data)

# huffmann encoding
codebook_22 = codebook_generator(data_221)
data_222 = huffman_encoding(data_221, codebook_22)
```

1.2.10 Diffrential encoding + RLE + Huffman encoding

[ ]:
```
# differential encoding
data_231 = diff_encoding(data)

# run-length encoding
data_232 = rl_encoding(data_231)

# huffman encoding
codebook_23 = codebook_generator(data_232)
data_233 = huffman_encoding(data_232, codebook_23)
```

2 Part 3: Image Coding

[22]:
```
# image
#img = 'sample.png'
img = 'Lhotse_Mountain_8-Bit_Grayscale.jpg'

# reading image / convert to numpy array
im = np.asarray(Pimage.open(img))
im = im.flatten()

# differential encoding
#data_31 = diff_encoding(im)

# run-length encoding
#data_32 = rl_encoding(data_31)

# huffman encoding
#codebook_3 = codebook_generator(data_32)
#code_3 = huffman_encoding(data_32, codebook_3)

freq_table = np.array(np.unique(im, return_counts=True))
freq_table = np.transpose(freq_table)
freq_table
#im.shape
#print(code_3.shape
```

[22]:
```
array([[   39,   180],
       [   40,  4215],
       [   41,   115],
       [   52,   262],
       [   53, 32136],
       [   54,   151],
       [   72,   270],
       [   73,  5760],
       [   74,   236],
       [   77,   521],
       [   78,  7365],
       [   79,   293],
       [  109,  1339],
       [  110, 25886],
       [  111,  1164],
       [  146,   795],
       [  147, 16518],
       [  148,   803],
       [  179,   809],
       [  180, 21597],
       [  181,   834],
```

```
       [  230,   943],
       [  231, 50148],
       [  232,  1660]])
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: