ECE 271A Statistical Learning 1

a) The reasonable estimate for the prior probability is the number of the training set of a specific class divided by the total number of training sets.

$$P_Y(cheetah) = \frac{size\ of\ foreground\ training\ samples}{size\ of\ total\ training\ samples}$$

$$P_Y(grass) = \frac{size\ of\ background\ training\ samples}{size\ of\ total\ training\ samples}$$

b)

- For this problem, random variable $X \in \{1,2,3,\dots,64\}$ is the index of the second largest coefficient of each $(8 * 8)$ image block. $P_{X|Y}(x|cheetah), P_{X|Y}(x|grass)$ can be represented as a $(64 * 1)$ vector where each element of the vector represents the probability of the random variable X equals the index of that element.
- To solve this problem, for foreground and background training samples, we will count the frequency of each index that happens to be the index of the second largest coefficient of one training sample. Then divide the frequency with the size of the foreground (background) training sample to transform frequency to probability.

c)

- Use an $(8 * 8)$ sliding window to convert a $(255 * 270)$ image matrix to $247 * 262$ $(8 * 8)$ image blocks.
- For each image block, compute the DCT and find the second largest coefficient a_{ij} within the block. Then transform the index (i, j) within the matrix to the index of a $(64 * 1)$ vector using the zig-zag file and store each index in an array A_matrix.
- For each element in A_matrix compute:

$$P_Y(cheetah) * P_{X|Y}\left(A_{matrix[i]}\middle|cheetah\right) \qquad a$$

$$P_Y(grass) * P_{X|Y}\left(A_{matrix[i]}\middle|grass\right) \qquad b$$

  If the $a \geq b$ then then the image block represented by $A_{matrix[i]}$ should be labeled as foreground, which in our case is 1. Otherwise it should be labeled as background, which in our case is 0. Store the label of image block in a numpy array.
- Reshape the array as a $(247 * 262)$ so that the we can visualize the result.

d)

- First we need to do some paddings around the picture so as to maintain the size of the picture, which is (255,270).
- Transform the image 'cheetah mask.bmp' as a numpy array transform the scale the numpy array to 1 (Divide by 255).
- $P_{error} = \frac{number\ of\ mistakes}{size\ of\ the\ picture}$

ECE 271A HW #1 The Cheetah Problem

The first step is to load the traning data with .mat format. For Pyhton I use package scipy.io.loadmat.

In [1]:

```
#import data
import numpy as np
from scipy.io import loadmat
m = loadmat('homework1/TrainingSamplesDCT_8.mat')
```

In [2]:

```
m
```

Out[2]:

```
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN, Created on: Tue
Sep 30 09:32:20 2003',
 '__version__': '1.0',
 '__globals__': [],
 'TrainsampleDCT_FG': array([[1.62254902e+00, 4.38433862e-01, 1.994701
28e-01, ...,
         8.49984789e-03, 3.78270042e-03, 3.02880400e-03],
        [1.56372549e+00, 8.21143328e-02, 9.09413795e-02, ...,
         3.18521582e-03, 5.81148077e-03, 8.88559648e-03],
        [1.24607843e+00, 1.06458077e-01, 4.86748243e-02, ...,
         1.98793183e-03, 2.02433826e-03, 2.26600748e-03],
        ...,
        [8.50000000e-01, 2.13586353e-02, 2.73332331e-02, ...,
         5.28559315e-03, 5.74533250e-04, 1.30883711e-03],
        [1.32696078e+00, 1.27736043e-02, 7.25986937e-02, ...,
         2.48390433e-03, 5.72821344e-03, 6.14234163e-04],
        [1.40637255e+00, 5.37446031e-02, 3.52432448e-02, ...,
         1.16665737e-02, 7.05600416e-04, 6.92833289e-04]]),
 'TrainsampleDCT_BG': array([[2.79215686e+00, 1.82403883e-01, 8.223819
82e-02, ...,
         8.99171356e-04, 8.20162208e-04, 5.56256629e-03],
        [2.77352941e+00, 1.89948302e-01, 7.13763275e-01, ...,
         8.71660205e-03, 7.92740281e-04, 2.27229935e-03],
        [2.80147059e+00, 1.74993685e-02, 4.21107915e-02, ...,
         1.37963126e-03, 1.62882836e-03, 2.88076472e-03],
        ...,
        [1.85000000e+00, 1.57010159e-01, 1.52944398e-02, ...,
         1.29102016e-03, 6.34867561e-05, 1.92029521e-03],
        [2.00735294e+00, 6.66499678e-02, 7.33864684e-02, ...,
         2.22899787e-03, 3.96008531e-03, 3.22405038e-04],
        [2.57205882e+00, 6.14495544e-02, 9.93700379e-02, ...,
         5.52675439e-03, 1.34849406e-03, 9.29894006e-04]])}
```

The .mat file is loaded as a dictionary in Python;
For this problem, we will simply extract TrainsampleDCT_FG and TrainsampleDCT_BG out from the dictionary.

In [3]:

```
foreground,background = m['TrainsampleDCT_FG'],m['TrainsampleDCT_BG']
```

Calculate the prior probability for cheetah(foreground) and grass(background);
Here the reasonable estimate for the prior is to use total number of training samples to devide the number of training samples of the foreground or background.

In [4]:

```
total = foreground.shape[0] + background.shape[0]
prior_cheetah = foreground.shape[0] / total
prior_grass = background.shape[0] / total
print(prior_cheetah)
print(prior_grass)
```

```
0.1918649270913277
0.8081350729086723
```

In [5]:

```
#initialization
freq_FG,freq_BG = np.zeros(64),np.zeros(64)
FG,BG = [],[]
```

For each foreground and background training sample, find the index of the second largest coefficient. Store the coefficient in 'FG' and 'BG' and add 1 to specific index of the foreground and background frequency map.

In [6]:

```
for i in range(len(foreground)):
    temp = np.argsort(foreground[i])
    freq_FG[temp[-2]] += 1
    FG.append(temp[-2])
```

In [7]:

```
for i in range(len(background)):
    temp = np.argsort(background[i])
    freq_BG[temp[-2]] += 1
    BG.append(temp[-2])
```

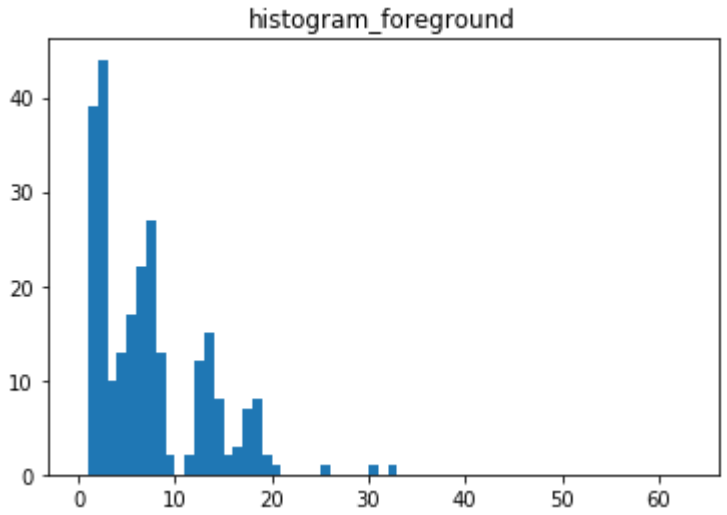
In [8]:

```
FG = np.array(FG)
BG = np.array(BG)
```

Histogram

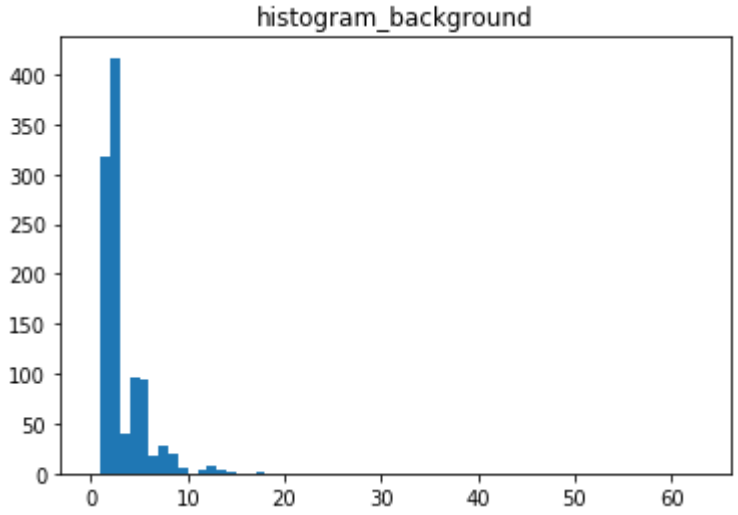
In [10]:

```
from matplotlib import pyplot as plt
plt.hist(FG, bins =  [i for i in range(64)])
plt.title("histogram_foreground")
plt.show()
```



In [11]:

```
plt.hist(BG, bins =  [i for i in range(64)])
plt.title("histogram_background")
plt.show()
```



Convert frequency to probability.

In [12]:

```
prob_FG = freq_FG / len(foreground)
prob_BG = freq_BG / len(background)
```

In [13]:

```
import imageio
im = imageio.imread('homework1/cheetah.bmp')
im_array = np.array(im)
```

Define a 2 dimensional DCT function using the 1 dimensional DCT function provided by scipy.

In [14]:

```
import scipy.fftpack
def dct2d(a):
    return scipy.fftpack.dct( scipy.fftpack.dct( a, axis=0, norm='ortho' ), axis=1,
```

In [15]:

```
zig_zag = np.array([[0,1,5,6,14,15,27,28],[2,4,7,13,16,26,29,42],[3,8,12,17,25,30,41
                    [9,11,18,24,31,40,44,53],[10,19,23,32,39,45,52,54],[20,22,33,38,4
                    [21,34,37,47,50,56,59,61],[35,36,48,49,57,58,62,63]])
zz_flat = zig_zag.flatten()
print(zz_flat)
```

```
[ 0  1  5  6 14 15 27 28  2  4  7 13 16 26 29 42  3  8 12 17 25 30 41
43
  9 11 18 24 31 40 44 53 10 19 23 32 39 45 52 54 20 22 33 38 46 51 55
60
 21 34 37 47 50 56 59 61 35 36 48 49 57 58 62 63]
```

Convert the (255,270) matrix to blocks of size (8,8), calculate the DCT of each block and get the index of the second largest coefficient (using zig-zag map);
For each block, calcuate probability that this block is whether a foreground or a background using BDT. Then label the block with the state that has a higher probability and store the state of each block in A.

In [16]:

```
A = []
for i in range(0,len(im_array)-8):
    for j in range(0,im_array.shape[1]-8):
        row_start,row_end = i,i+8
        col_start,col_end = j,j+8
        block = im_array[row_start:row_end,col_start:col_end]
        block_dct = dct2d(block).flatten()
        index = np.argsort(block_dct)[-2]
        if prior_cheetah * prob_FG[zz_flat[index]] >= prior_grass * prob_BG[zz_flat[
            A.append(1)
        else:
            A.append(0)
A = np.array(A)
print(A)
```

```
[0 0 1 ... 0 0 1]
```

In [17]:

```
A_matrix = np.reshape(A,(247,262))
```

Right now the size of the matrix(picture) is (247,262), we need to do some paddings around the picture so as to maintain the size of the picture, which is (255,270).

In [18]:

```
A_matrix_padding = np.lib.pad(A_matrix,(4,4),'constant',constant_values = 0)
```

In [19]:

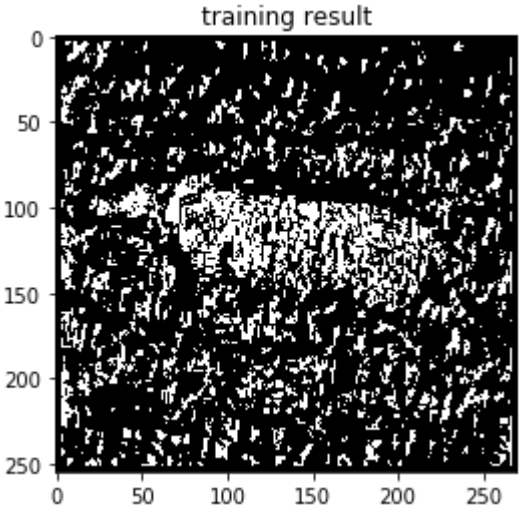
```
A_matrix_padding.shape
```

Out[19]:

```
(255, 270)
```

In [20]:

```
import matplotlib.pyplot as plt
plt.imshow(A_matrix_padding,cmap='gray')
plt.title("training result")
plt.show()
```



In [21]:

```
# store the test data as a numpy array
im_test = imageio.imread('homework1/cheetah_mask.bmp')
im_test_array = np.array(im_test)
# convert 255 to 1 for error calculation
im_test_array = im_test_array / 255
```

In [22]:

```
a = np.absolute(im_test_array - A_matrix_padding)
```

In [23]:

```
prob_error = np.sum(a) / (255 * 270)
```

In [24]:

```
prob_error
```

Out[24]:

```
0.24422657952069718
```