Tp5 Dimitris

Non-blind watermark detection

Exercise 1

Read the image cameraman.tif in. It will serve as host image x. For given hypothesis:

$$H0: v = x + z$$

$$H1: v = x + w + z$$

where x is the host image, v is the marked image, w is the watermark and z is additive white Gaussian noise, i.e.$Z \sim N(0, \sigma^2_{noise} I)$

```
import matplotlib.image as mpimg; import matplotlib.pyplot as plt; import numpy

url = "https://github.com/partizanos/multimedia_security/raw/master/multimedia_
response = requests.get(url)
im = Image.open(BytesIO(response.content))
X  = np.array(im)
N1, N2 = X.shape
N = int(N1*N2)
```

Hypothesis and watermark definition as on TP4

```
def imageShape(img):
  N1, N2 = X.shape
  assert X.shape == (256, 256)
  return int(N1*N2)


def watermark(gamma, img, theta):
  assert type (gamma) ==int
  N1, N2 = img.shape
  size = imageShape(img)
  assert type (size) ==int
  indices=[]

  #   Generate a matrix w’ of size x with uniform distributed values {−1, 1}.
  #   The magnitude of these two values governs the watermark strength.
  w_prime = np.random.uniform(-gamma,gamma,size)
  w_prime = w_prime.reshape((N1, N2))

  #   Randomly sample from matrix w’ with a given density θN = 0.5
  for i in range(int(size * theta )):
    indices.append((np.random.randint(N1), np.random.randint(N2)))

  w = np.zeros((N1,N2))
  for tup in indices:
    w[tup] = w_prime[tup]
  return w



def get_Hypothesis(X, sigma_noise, gamma, theta):
  # V = X + Z
  N1, N2 = X.shape
  Z = np.random.normal(0, sigma_noise, imageShape(X)).reshape((N1, N2))
  V_H0 = X + Z
  w = watermark(gamma, X, theta)
```

```
  V_H1 = V_H0 + w
  return V_H0, V_H1, w

def rho_non_blind(v, w, X, N ):
  w_non_blind = v - X
  return  np.sum(w.reshape((N))*w_non_blind.reshape((N))) / (N)

```

For hypothesis H_1 and ρ^{H_1} the watermark w is generated J times with a fixed strength γ = ±1 2 and a fixed density θ_N = 0.1. The noise realization z is again fixed with $\sigma_{noise} = 50$.

Fill out Table 1 with all results. Note that obviously only the noise and not the watermark has influence on hypothesis H 0 , so the relevant cells have been grayed out.

Determine rho mu and sigma parameter for H_0 and H_1

```
# sigmas =[50, 100];
# densities = [0.1, 0.3];
# gammas = [1, 5];

# sigma_noise = 50
# theta = 0.1
# gamma = 1


def experiment(X, sigma_noise, gamma, theta, J ):
  rho_h0_list = []
  rho_h1_list = []

  for j in range(J):
    vh0, vh1, w = get_Hypothesis(X, sigma_noise, gamma, theta)
    rho_h0_list.append(rho_non_blind(vh0, w, X, N ))
    rho_h1_list.append(rho_non_blind(vh1, w, X, N ))

  rho_h0_list = np.array(rho_h0_list)
  rho_h1_list = np.array(rho_h1_list)

  mu_h0 = np.sum(rho_h0_list)/J
  mu_h1 = np.sum(rho_h1_list)/J

  sigma_h0 = (np.sum(rho_h0_list - mu_h0)**2) / J
  sigma_h1 = (np.sum(rho_h1_list - mu_h1)**2) / J

  print("sigma noise: ", sigma_noise)
  print("theta: ", theta)
  print("gamma: ", gamma)
  print("### mu|h0: ",mu_h0, ", : mu|h1 ", mu_h1, " sigma_h0: ", sigma_h0, "si
  return mu_h0, mu_h1,  sigma_h0,  sigma_h1
```

```
sigma_noise, gamma, theta = 50, 5, 0.3
vh0, vh1, w = get_Hypothesis(X, sigma_noise, gamma, theta)

fig, ax = plt.subplots(
        ncols=2,
        sharex=True,
        sharey=True,
        figsize=(18, 18)
)

ax[0].imshow(vh0, 'gray')
ax[0].set_title("h0")
ax[1].imshow(vh0, 'gray')
ax[1].set_title("h1")
```

Calculate and display the Receiver Operating Characteristic (ROC) curve for the binary threshold test following the above mentioned experiment set up. The detection threshold is denoted with $T_{\rho\_non-blind}$.

```
from scipy.stats import norm
sigma_noise, gamma, theta = 100, 5, 0.3

[mu_H0, mu_H1, var_H0, var_H1] =  experiment(X, sigma_noise, gamma, theta, J )
threshold = np.linspace(mu_H0-var_H0,mu_H1+var_H1,12);
PFA=[]
PMISS=[]
for i in range(len(threshold)):
    PFA.append(1 - norm.cdf(threshold[i], mu_H0, var_H0))
    PMISS.append(norm.cdf(threshold[i], mu_H1, var_H1))

plt.plot(PFA); plt.show(); plt.title("PFA")
plt.plot(PMISS); plt.show(); plt.title("PMISS")
plt.plot(PFA,PMISS); plt.show(); plt.title("ROC")
```

```

sigmas =[50, 100];
densities = [0.1, 0.3];
```

```
gammas = [1, 5];
J = 100
for sigma in sigmas:
  for theta in densities:
    for gamma in gammas:
      print("Running experiment with parameters: ", sigma, theta, gamma)
      mu_h0, mu_h1,  sigma_h0,  sigma_h1 = experiment(X, sigma, gamma, theta, 

```

What can you conclude about non-blind watermark detection given the strength of the watermark and the noise variance?

Answer

- For higher values of theta watermark mu is significantly higher.
- For sigma 100 the standarad deviation was 2 orders of magnitude higher tha for sigma 50.

Double-click (or enter) to edit