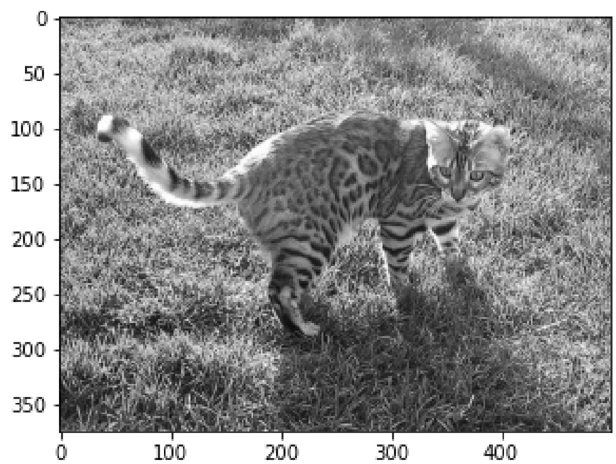
Exercise 2

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import csv
```

In [ ]:
```
path = "C:\\Users\\Parth\\OneDrive - purdue.edu\\Spring 2023\\ECE ML\\Homework\\HW3\\hom

train_cat = np.matrix(np.loadtxt(path + 'train_cat.txt', delimiter = ','))
train_grass = np.matrix(np.loadtxt(path + 'train_grass.txt', delimiter = ','))
```

In [ ]:
```
Y = plt.imread(path + 'cat_grass.jpg') / 255
plt.imshow(Y,cmap='gray')
print(f"Dimensions of the image are {Y.shape}")
```

```
Dimensions of the image are (375, 500)
```



In [ ]:
```
train_cat = []

with open(path + 'train_cat.txt') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')

    for data in csv_reader:
      train_cat.append(data)

train_cat = np.array(train_cat, dtype='float64')
train_cat = train_cat
K1 = train_cat.shape[1]
print(f"K1 = {K1}")
```

```
K1 = 1976
```

In [ ]:
```
train_grass = []

with open(path + 'train_grass.txt') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')

    for data in csv_reader:
      train_grass.append(data)

train_grass = np.array(train_grass, dtype='float64')
```

```
K0 = train_grass.shape[1]
print(f"K0 = {K0}")
```

```
K0 = 9556
```

In [ ]:
```
pi_1 = K1/(K1 + K0)
pi_0 = 1 - pi_1

print(f"pi_0 = {pi_0:.2f}, pi_1 = {pi_1:.2f}")
```

```
pi_0 = 0.83, pi_1 = 0.17
```

In [ ]:
```
mu_0 = np.sum(train_grass, axis=1)/train_grass.shape[1]
mu_1 = np.sum(train_cat, axis=1)/train_cat.shape[1]
mu_0 = mu_0.reshape(64,1)
mu_1 = mu_1.reshape(64,1)


print(f"The first few values in mu_0 are: {mu_0[:2]}")
print(f"The first few values in mu_1 are: {mu_1[:2]}")
```

```
The first few values in mu_0 are: [[0.48249575]
 [0.4864399 ]]
The first few values in mu_1 are: [[0.44080734]
 [0.43871359]]
```

In [ ]:
```
sigma_0 = np.zeros((64, 64))
for i in train_grass.T:
  j = i.reshape((64,1))
  temp = np.reshape(j - mu_0, (64,1))
  sigma_0 += np.matmul(temp, temp.T)
sigma_0 = sigma_0/(train_grass.shape[1] - 1)

sigma_1 = np.zeros((64, 64))
for i in train_cat.T:
  j = i.reshape((64,1))
  temp = np.reshape(j - mu_1, (64,1))
  sigma_1 += np.matmul(temp, temp.T)
sigma_1 = sigma_1/(train_cat.shape[1] - 1)
```

In [ ]:
```
print(f"The first few values in sigma_0 are: \n{sigma_0[:2,:2]}")
print(f"The first few values in sigma_1 are: \n{sigma_1[:2,:2]}")
```

```
The first few values in sigma_0 are:
[[0.064484   0.0369168 ]
 [0.0369168  0.06623457]]
The first few values in sigma_1 are:
[[0.04307832 0.03535405]
 [0.03535405 0.0424875 ]]
```

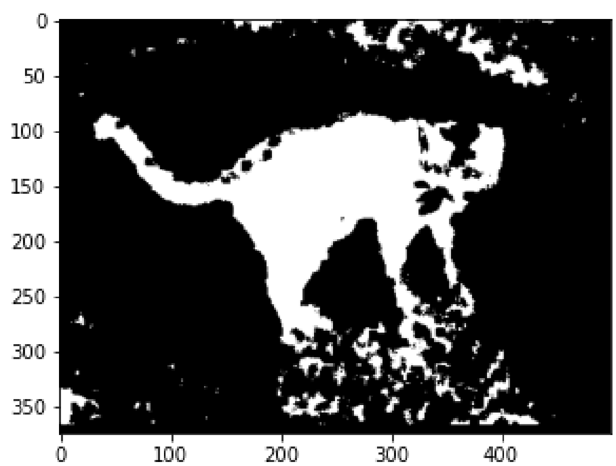
In [ ]:
```
sig_0_inv = np.linalg.inv(sigma_0)
sig_1_inv = np.linalg.inv(sigma_1)
sig_0_det = np.linalg.det(sigma_0)
sig_1_det = np.linalg.det(sigma_1)
```

In [ ]:
```
truth = plt.imread(path + 'truth.png')
truth = np.round(np.array(truth, dtype='float64'))
Beta = 0
Alpha = 0
```

```
Tot_P = np.count_nonzero(truth == 1)
Tot_N = np.count_nonzero(truth == 0)

M,N = Y.shape
mask = np.zeros(Y.shape)
for i in range(0,M-8):
  for j in range(0,N-8):
    block = Y[i:i+8, j:j+8] # This is a 8x8 block
    block = block.flatten()
    block = block.reshape(64,1)
    LHS = -0.5*np.matmul(np.matmul((block - mu_1).T, sig_1_inv),(block - mu_1)) + np.log
    RHS = -0.5*np.matmul(np.matmul((block - mu_0).T, sig_0_inv),(block - mu_0)) + np.log
    mask[i,j] = 1 if (LHS > RHS) else 0

    # TP, FP
    if (mask[i,j] == 1 and truth[i,j] == 1):
      Beta +=1

    if (mask[i,j] == 1 and truth[i,j] == 0):
      Alpha +=1

Beta = Beta/Tot_P
Alpha = Alpha/Tot_N
```

In [ ]:
```
plt.imshow(mask, cmap='gray')
```

Out[ ]: <matplotlib.image.AxesImage at 0x20233098cf8>

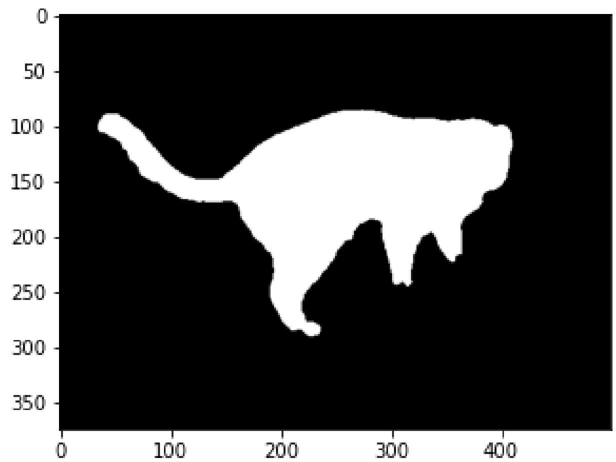


In [ ]:
```
M,N = mask.shape
MAE = np.sum(abs(truth[0:M-8, 0:N-8]-mask[0:M-8, 0:N-8]))/(M*N)
MAE
```

Out[ ]: 0.08764266666666666

In [ ]:
```
plt.imshow(truth, cmap='gray')
```

Out[ ]: <matplotlib.image.AxesImage at 0x286512edc50>

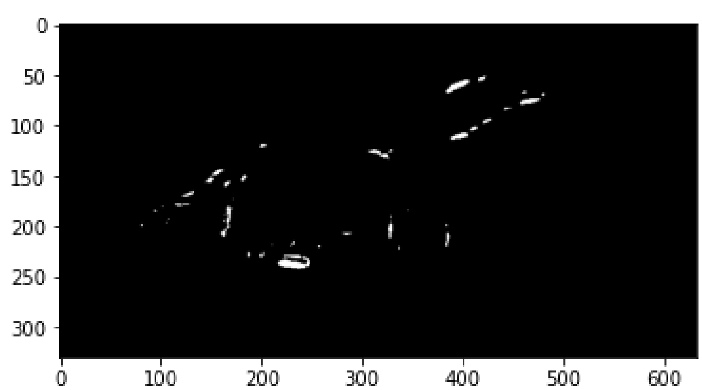


Test image

In [ ]:
```
from cv2 import cvtColor, COLOR_BGR2GRAY, imread

test = imread(path + 'doge3.jpg')
test = np.round(np.array(test, dtype='float32'))
test = cvtColor(test, COLOR_BGR2GRAY)

M,N = test.shape
mask = np.zeros(test.shape)
for i in range(0,M-8):
  for j in range(0,N-8):
    block = test[i:i+8, j:j+8] # This is a 8x8 block
    block = block.flatten()
    block = block.reshape(64,1)
    LHS = -0.5*np.matmul(np.matmul((block - mu_1).T, sig_1_inv),(block - mu_1)) + np.log
    RHS = -0.5*np.matmul(np.matmul((block - mu_0).T, sig_0_inv),(block - mu_0)) + np.log
    mask[i,j] = 1 if (LHS > RHS) else 0
```

In [ ]:
```
plt.imshow(mask, cmap='gray')
```

Out[ ]: <matplotlib.image.AxesImage at 0x202337de1d0>



In [ ]:
```
plt.imshow(test, cmap='gray')
```

Out[ ]: <matplotlib.image.AxesImage at 0x2023383ec88>

Exercise 3

Likelihood Ratio Test

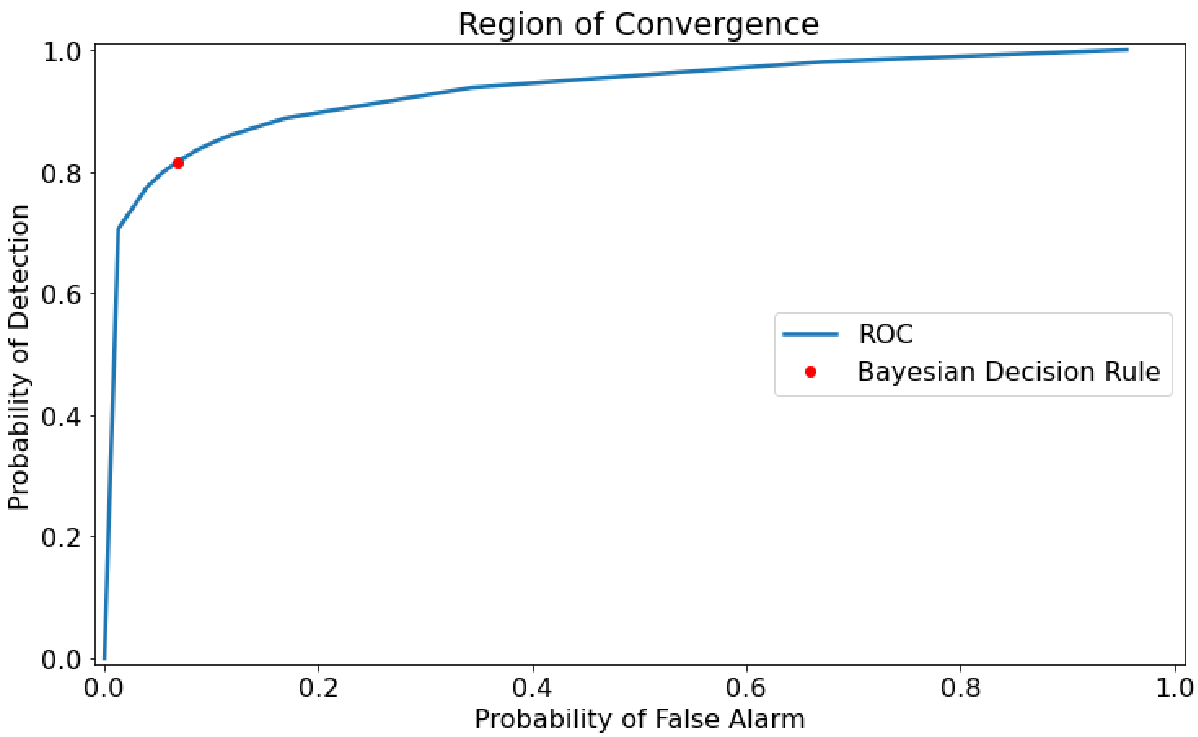
In [ ]:
```
tau_array = np.log(np.logspace(-1000,1000, base=np.e, num=20))
tau_array = np.array([-1000, -500,  -100, -50, -20,  -10,    -9,    -8,    -7,    -6,
           -3,    -2,    -1,     0,     1,     2,     3,     4,     5,
            6,     7,     8,     9, 10, 20, 50, 100, 500,  1000])
Beta_array = np.zeros(len(tau_array))
Alpha_array = np.zeros(len(tau_array))

M,N = Y.shape
mask = np.zeros(Y.shape)
for tau in range(len(tau_array)):
  for i in range(0,M-8):
    for j in range(0,N-8):
      block = Y[i:i+8, j:j+8] # This is a 8x8 block
      block = block.flatten()
      block = block.reshape(64,1)
      LHS = -0.5*np.matmul(np.matmul((block - mu_1).T, sig_1_inv),(block - mu_1)) - 0.5*
      RHS = -0.5*np.matmul(np.matmul((block - mu_0).T, sig_0_inv),(block - mu_0)) - 0.5*
      mask[i,j] = 1 if (LHS - RHS>= tau_array[tau]) else 0

      # TP, FP
      if (mask[i,j] == 1 and truth[i,j] == 1):
        Beta_array[tau] +=1

      if (mask[i,j] == 1 and truth[i,j] == 0):
        Alpha_array[tau] +=1

Beta_array = Beta_array/Tot_P
Alpha_array = Alpha_array/Tot_N
```

```
c:\Users\Parth\miniconda3\envs\TF\lib\site-packages\numpy\core\function_base.py:265: Run
timeWarning: overflow encountered in power
  return _nx.power(base, y)
c:\Users\Parth\miniconda3\envs\TF\lib\site-packages\ipykernel_launcher.py:1: RuntimeWarn
ing: divide by zero encountered in log
  """Entry point for launching an IPython kernel.
```

In [ ]:
```
import matplotlib
matplotlib.rcParams.update({'font.size': 16})
plt.figure(figsize=(12,7))
```

```
plt.plot(Alpha_array, Beta_array, linewidth=2.5)
plt.plot(Alpha, Beta, 'o',c='r')
plt.title("Region of Convergence")
plt.xlabel("Probability of False Alarm")
plt.ylabel("Probability of Detection")
plt.xlim(-0.01,1.01)
plt.ylim(-0.01,1.01)
plt.legend(["ROC","Bayesian Decision Rule"], loc=5)
```

Out[ ]: <matplotlib.legend.Legend at 0x28656ee04e0>



Linear Regression

In [ ]:
```
X1 = train_cat.T
X0 = train_grass.T
A = np.vstack((X1,X0))

y1 = [1 for i in range(X1.shape[0])]
y0 = [-1 for i in range(X0.shape[0])]
y = np.array(y1 + y0)
b = y.reshape((y.shape[0],1))

t1 = np.linalg.inv(np.matmul(A.T,A))
t2 = np.matmul(A.T, b)
theta = np.matmul(t1, t2)
```

In [ ]:
```
tau_array = np.array([-1000, -500,  -100, -50, -20,  -10,    -9,    -8,    -7,    -6,
           -3,    -2,    -1,     0,     1,     2,     3,     4,     5,
            6,     7,     8,     9, 10, 20, 50, 100, 1000])
Beta_array = np.zeros(len(tau_array))
Alpha_array = np.zeros(len(tau_array))

M,N = Y.shape
mask = np.zeros(Y.shape)
```

```
for tau in range(len(tau_array)):
  for i in range(0,M-8):
    for j in range(0,N-8):
        block = Y[i:i+8, j:j+8] # This is a 8x8 block
        block = block.flatten()
        block = block.reshape(64,1)
        test = np.matmul(theta.T, block)
        mask[i,j] = 1 if (test >= tau_array[tau]) else 0

        # TP, FP
        if (mask[i,j] == 1 and truth[i,j] == 1):
            Beta_array[tau] +=1

        if (mask[i,j] == 1 and truth[i,j] == 0):
            Alpha_array[tau] +=1

Beta_array = Beta_array/Tot_P
Alpha_array = Alpha_array/Tot_N
```

In [ ]:

```
import matplotlib
matplotlib.rcParams.update({'font.size': 16})
plt.figure(figsize=(12,7))
plt.plot(Alpha_array, Beta_array, linewidth=2.5)
plt.title("Region of Convergence")
plt.xlabel("Probability of False Alarm")
plt.ylabel("Probability of Detection")
plt.xlim(-0.01,1.01)
plt.ylim(-0.01,1.01)
```

Out[ ]: (-0.01, 1.01)