Міністерство освіти і науки України

НТУУ «Київський політехнічний інститут ім. Ігоря Сікорського»

Фізико-технічний інститут

Лабораторна робота №2

З дисципліни

«Автоматизація обробки ІзОД»

Варіант 6

**Виконав:**

Студент 5 курсу ФТІ

групи ФЕ-91мп

Назаров О.В.

**Перевірив:**

Прогонов Д. О.

Київ-2020

***Вихідні дані***

Тестовий пакет – MIRFlickr-20k (https://press.liacs.nl/mirflickr/#sec\_download)

Вибірка зображень – 250 зображень;

Формування вибірки зображень – псевдовипадкове, з використанням генератора Мерсена (стартове значення співпадає з номером студента в загальному списку групи) за модулем кількості зображень в тестовому пакеті.

***Лабораторна робота №2***

1. Сформувати тестову вибірку зображень з вихідного пакета;
2. Для кожного каналу кольору кожного зображення з тестового пакета обчислити наступні характеристики:
   1. Математичне сподівання і дисперсію;
   2. Коефіцієнти асиметрії та ексцесу (нормалізований);
3. Побудувати вектори параметрів зображень, що складаються з:
   1. Математичних очікувань значень яскравості для кожного каналу кольору;
   2. Математичних очікувань і дисперсії значень яскравості для кожного каналу кольору;
   3. Математичних очікувань, дисперсії і коефіцієнта асиметрії значень яскравості для кожного каналу кольору;
   4. Математичних очікувань, дисперсії, коефіцієнтів асиметрії та ексцесу значень яскравості для кожного каналу кольору;
4. Побудувати гаусові моделі зображень з використанням розрахованих раніше параметрів.
5. Провести декомпозицію кожного каналу кольору кожного зображення з застосуванням методу головних компонент (PCA):
   1. Варіюючи кількість компонент, провести реконструкцію окремих каналів кольору зображень (від компонент з найбільшою енергією поступово переходячи до компонентів з мінімальною енергією).
   2. Побудувати залежність помилки відновлення (середнє відхилення вихідного зображення відреконструйованого, MSE) від кількості використаних компонент.
6. Провести моделювання окремих каналів кольору зображень з використанням марковських ланцюгів:
   1. Для кожного каналу кольору кожного зображення розрахувати стохастическую матрицю марковської ланцюга першого і другого порядків (обробка пікселів по горизонталі справа наліво і навпаки, а також по вертикалі зверху вниз і навпаки). У звіті привести явний вигляд однієї марковської ланцюга для одного з каналів кольору тестового зображення;
   2. Перевірити властивість регулярності, реккурентное і незворотності (irreducible) для отриманих марковских моделей для 5 ітерацій.

**ІI. Хід роботи**

Роботу виконуватимемо мовою Python за допомогою блокового інтерпритатора Jupyter. Також в роботі будкть використані такі бібліотеки як:

* Os
* Matplotlib
* Numpy
* Scipy
* Pandas
* Sklearn
* Seaborn
* CV2
* networkx

1.Завантаження вибірки зображень відносно варіанту

test\_index = random.sample(range(6,25001),250)

# Path to where my test images are stored

img\_folder = os.path.join(os.getcwd(), 'mirflickr/')

test\_images = []

for i in test\_index:

test\_images.append('C:\\Users\\malin\\Desktop\\Diplom\\datasets\\mirflickr/im'+ str(i) + '.jpg')

def convert(filename):

im = Image.open(filename,mode = "r")

return np.array(im)

def img\_list():

t = time.time()

np\_arrays = map(convert, test\_images)

arr\_list = [i for i in np\_arrays]

return arr\_list

2.Створення датафрейму з данними математичних очікувань, дисперсії, коефіцієнтів асиметрії та ексцесу значень яскравості для кожного каналу кольору.

params = ['Image Number','Channel Number','ExpValue(UnifDist)','Variance','Skew',"Kurtosis","Intensity of channel",'Img']

def part1(img):

param\_list = []

for i in range(len(img)):

temp = img[i]# інтенсивність яскравості кожного каналу

mn = temp.sum(axis=0).sum(axis=0)/(temp.shape[0]\*temp.shape[1])

intensity\_of\_channels = mn/np.linalg.norm(mn, ord=None)

for Channel in range(3):

img\_ch = img[i][:,:,Channel]

param\_list.append(list((test\_index[i],Channel,np.mean(img\_ch),np.var(img\_ch)

,skew(img\_ch,None),kurtosis(img\_ch,None),intensity\_of\_channels[Channel],img\_ch)))

return pd.DataFrame(param\_list,columns = params)

images = img\_list()

Data = part1(images)

Створюємо масив даних для кожного каналу кольору

x\_i\_RED = Data.loc[Data['Channel Number'] == 0][['ExpValue(UnifDist)','Variance','Skew','Kurtosis','Intensity of channel']]

x\_i\_GREEN = Data.loc[Data['Channel Number'] == 1][['ExpValue(UnifDist)','Variance','Skew','Kurtosis','Intensity of channel']]

x\_i\_BLUE = Data.loc[Data['Channel Number'] == 2][['ExpValue(UnifDist)','Variance','Skew','Kurtosis','Intensity of channel']]

labels = x\_i.columns

Будуємо коваріаційну матрицю і матрицю математичних сподівань по параметрам

x\_i\_RED = Data.loc[Data['Channel Number'] == 0][['ExpValue(UnifDist)','Variance','Skew','Kurtosis','Intensity of channel']]

x\_i\_GREEN = Data.loc[Data['Channel Number'] == 1][['ExpValue(UnifDist)','Variance','Skew','Kurtosis','Intensity of channel']]

x\_i\_BLUE = Data.loc[Data['Channel Number'] == 2][['ExpValue(UnifDist)','Variance','Skew','Kurtosis','Intensity of channel']]

labels = x\_i.columns

B = np.zeros((3,1,5))#mat expectation

expec\_RED = np.mean(x\_i\_RED)

expec\_GREEN = np.mean(x\_i\_GREEN)

expec\_BLUE = np.mean(x\_i\_BLUE)

B[0],B[1],B[2] = expec\_RED,expec\_GREEN,expec\_BLUE

A = np.zeros((3,5,5)) # covariance matrix

cov\_diag\_RED = np.eye(5,5) \* np.array(x\_i\_RED.var())

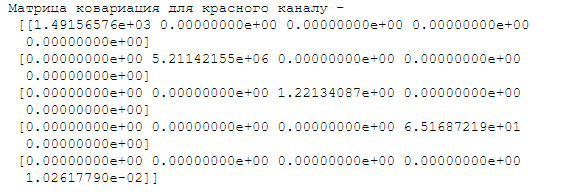
cov\_diag\_GREEN = np.eye(5,5) \* np.array(x\_i\_GREEN.var())

cov\_diag\_BLUE = np.eye(5,5) \* np.array(x\_i\_BLUE.var())

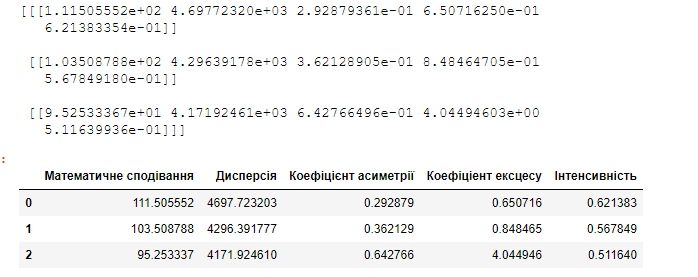
A[0],A[1],A[2] = cov\_diag\_RED,cov\_diag\_GREEN,cov\_diag\_BLUE

print(A[0])

Приклад матриці коваріації для одного із каналів



Вектори математичного очікування



4. Будуємо pdf – функцію моделі зображень з використанням розрахованих раніше параметрів.

gauss\_pdf = multivariate\_normal.pdf(x\_i\_scaled,std,cov\_diag)

print(' Standart deviation :' ,std ,'\n' ,

"Covariance matrix :",'\n',cov\_diag )

print(gauss\_pdf)gauss\_pdf = multivariate\_normal.pdf(gauss\_samples\_from\_distributions,std,cov\_diag)

5. Провести декомпозицію кожного каналу кольору кожного зображення з застосуванням методу головних компонент (PCA):

* 1. Варіюючи кількість компонент, провести реконструкцію окремих каналів кольору зображень (від компонент з найбільшою енергією поступово переходячи до компонентів з мінімальною енергією).
  2. Побудувати залежність помилки відновлення (середнє відхилення вихідного зображення відреконструйованого, MSE) від кількості використаних компонент.

Уніфікуємо розміри зображень для PCA model (sklearn).

Вибираєм розміри найменшого зображення щоб мінімізувати втрату інформації .

images = Data['Img']

shape\_0 = min([i.shape[0] for i in images])

shape\_1 = min([i.shape[1] for i in images])

def resize(images,min\_x,min\_y):

resized = []

for i in images:

img\_rez = cv2.resize(i,(shape\_0,shape\_1),interpolation=cv2.INTER\_CUBIC)

resized.append(img\_rez)

return np.array(resized).reshape((-1,shape\_0,shape\_1))

перетворюємо набір в одновимірний до виду (n\_images,n\_features)

і стандартизуємо в межах 0 - 1

resized\_images = resize(images,shape\_0,shape\_1)

flatened\_images = resized\_images.reshape((750,shape\_0\*shape\_1))

standarting\_images = flatened\_images/255

def image\_PCA(images,energy):

model = PCA(energy)

model.fit(images)

tr\_images = model.transform(images)

return model,tr\_images

Будуємо репрезентації з вибраною енергією

image\_PCA(standarting\_images,200)

відбудовуємо деякий набір зображень із репрезентацій і вираховуємо MSE похибку між відбудованим і оригінальним

def mse(imageA, imageB):

err = np.sum((imageA.astype("float") - imageB.astype("float")) \*\* 2)

err /= float(imageA.shape[0] \* imageA.shape[1])

return err

some\_channel\_index = np.random.choice(750,50)

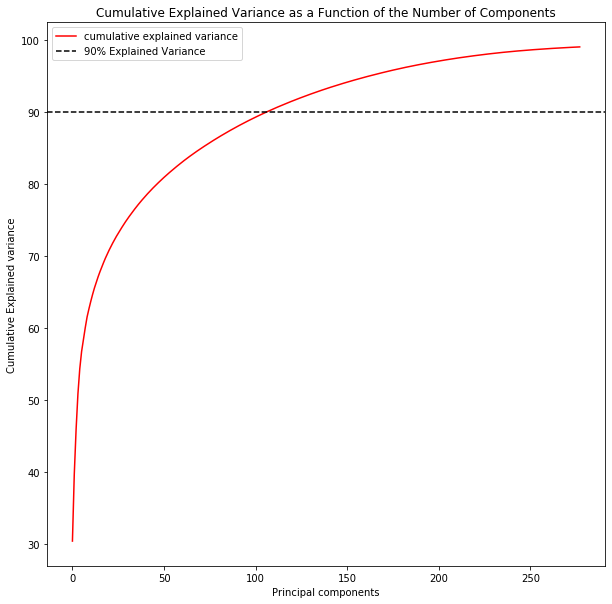
model,compressed\_img = image\_PCA(standarting\_images,.99)

reconstr = model.inverse\_transform(compressed\_img \* 255)

mse\_log = np.array(list(map(mse,reconstr.reshape((750,shape\_0,shape\_1)),standarting\_images.reshape((750,shape\_0,shape\_1)))))

print("Mean MSE - " ,np.mean(mse\_log))

Mean MSE - 5797.824853968864



Побудова графіку залежності MSE від кількості компонент

some\_channel\_index = np.random.choice(750,300)

some\_set = standarting\_images[some\_channel\_index]

some\_set.shape

n\_components = np.arange(50,300,25)

def func(n\_components,some\_set,shape\_0,shape\_1):

models\_data = []

compressed\_list\_by\_ncomponents = []

mse\_log\_list = []

mean\_mse = []

for i in n\_components:

model1,compressed\_images = image\_PCA(some\_set,i)

models\_data.append(model)

compressed\_list\_by\_ncomponents.append(compressed\_images)

reconstr = model1.inverse\_transform(compressed\_images)

mse\_log = np.array(list(map(mse,reconstr.reshape((300,shape\_0,shape\_1)),some\_set.reshape((300,shape\_0,shape\_1)))))

mse\_log\_list.append(mse\_log)

mean\_mse.append(np.mean(mse\_log))

return n\_components,np.array(mean\_mse),np.array(mse\_log\_list)

ar = func(n\_components,some\_set,shape\_0,shape\_1)

plt.figure(figsize=(10, 10))

plt.plot(ar[0],ar[1],color = 'red',label='dependence of mse from n\_components')

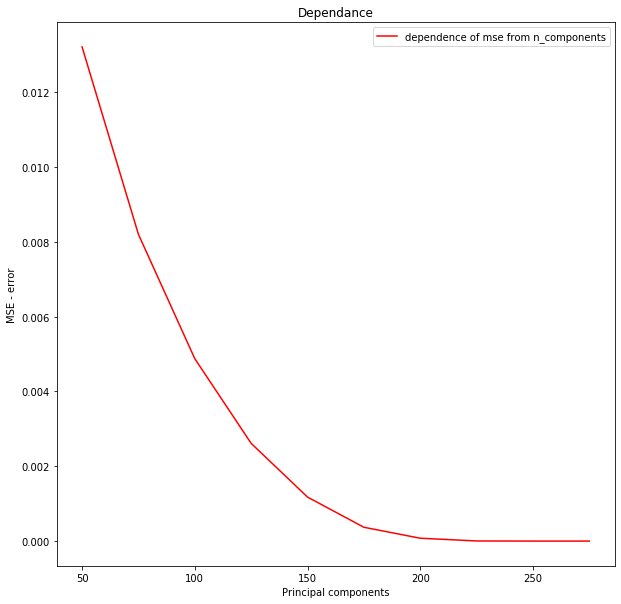
plt.title('Dependance')

plt.ylabel('MSE - error')

plt.xlabel('Principal components')

plt.legend(loc='best')

plt.show()



6. Провести моделювання окремих каналів кольору зображень з використанням марковських ланцюгів: a. Для кожного каналу кольору кожного зображення розрахувати стохастическую матрицю марковської ланцюга першого і другого порядків (обробка пікселів по горизонталі справа наліво і навпаки, а також по вертикалі зверху вниз і навпаки). У звіті привести явний вигляд однієї марковської ланцюга для одного з каналів кольору тестового зображення; b. Перевірити властивість регулярності, реккурентное і незворотності (irreducible) для отриманих марковских моделей для 5 ітерацій.

Функція для розрахування стохастичної матриці марковського ланцюга 1 і 2 порядків для кожного каналу кожного зображення

ima = np.array(img\_list())

resized\_images = resize(ima,shape\_0,shape\_1)

imas = resized\_images.reshape(250,167,226,3)

def Markov(images):

ar = []

for s in images:

for i in range(3):

arr = s[:,:,i].flatten()

prev\_color = arr[0]

markov\_matrix1 = np.zeros(shape=(256, 256))

for i in range(len(arr) - 1):

markov\_matrix1[arr[i]][arr[i + 1]] += 1

markov\_matrix = markov\_matrix1[0] / sum(markov\_matrix1[0])

for i in range(1, 256):

markov\_matrix = np.vstack((markov\_matrix, markov\_matrix1[i] / sum(markov\_matrix1[i])))

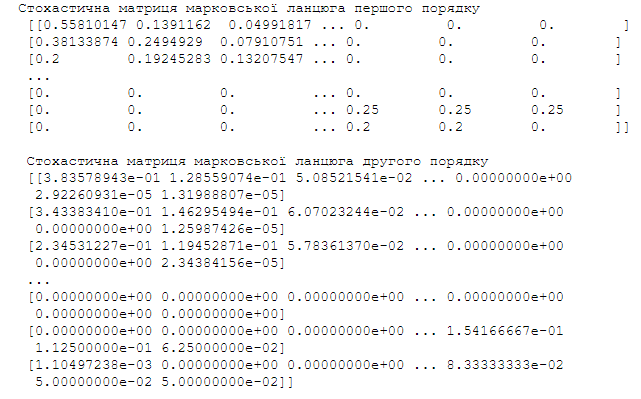
ar.append((markov\_matrix,np.linalg.matrix\_power(markov\_matrix, 2)))

return np.array(ar)

dat = Markov(imas)

print('Стохастична матриця марковської ланцюга першого порядку',"\n",dat[1][0])

print("\n",'Стохастична матриця марковської ланцюга другого порядку',"\n",dat[1][1])



Приклад марковського ланцюга

data = dat[0][0]

data = np.triu(data) + np.triu(data).T

ind = [str(i) for i in range(data.shape[0])]

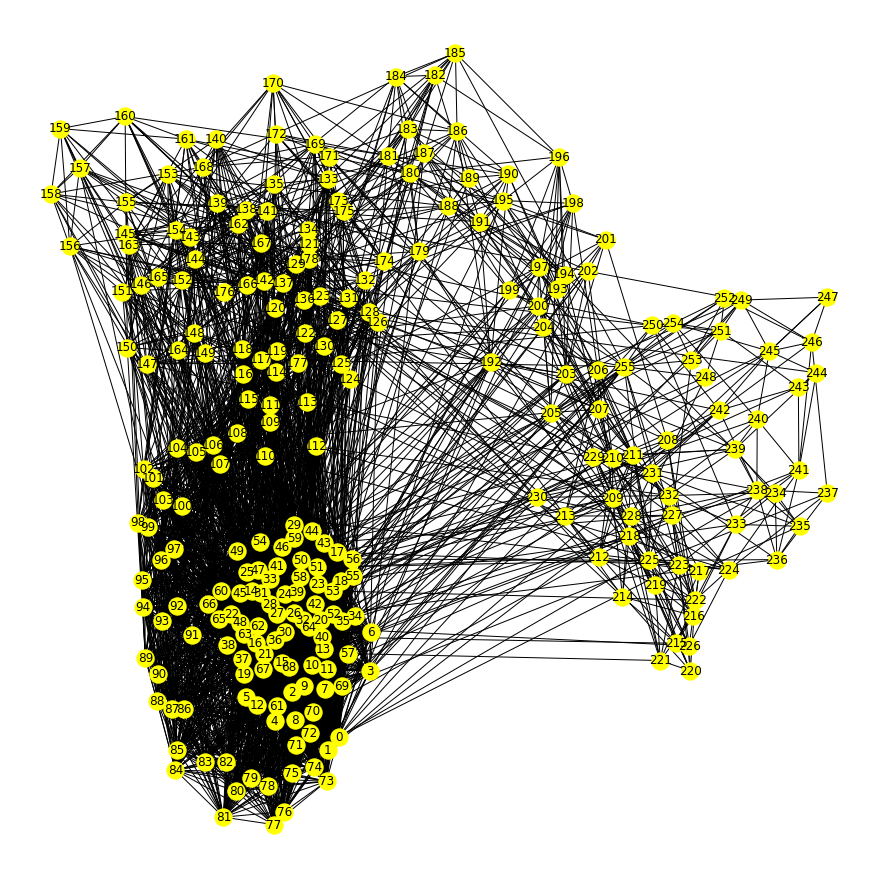
df2 = pd.DataFrame(data, index=ind, columns=ind)

plt.figure(1,figsize=(12,12))

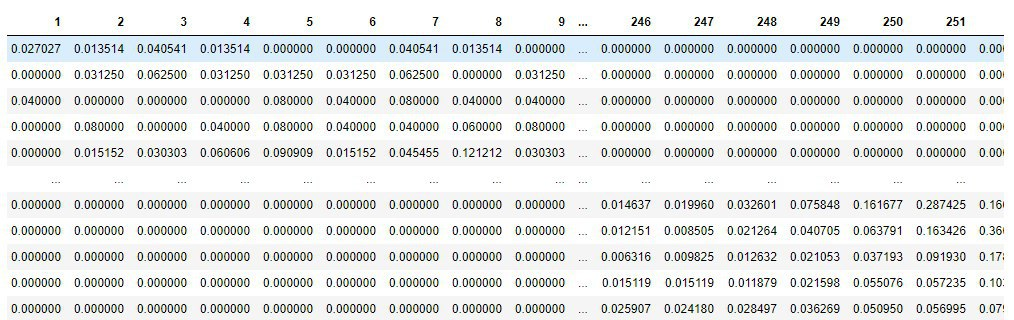
G2 = nx.from\_pandas\_adjacency(df2)

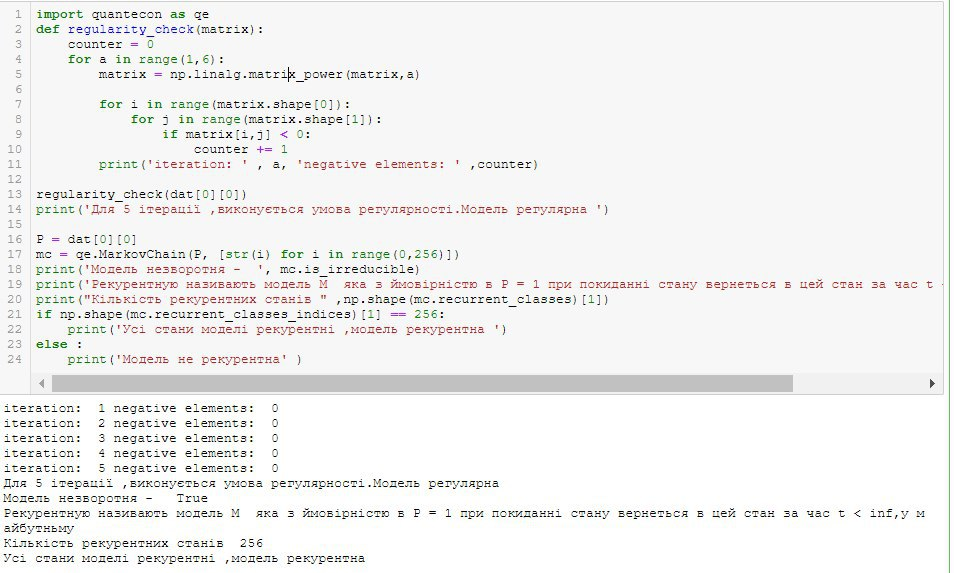
nx.draw(G2, with\_labels=True, node\_color='yellow')

plt.show()



Перевірка гіпотез



Матриця регулярна, незворотня і рекурентна .

ВИСНОВОК

В лабораторній роботі було проаналізовано вибірку з 250 зображень датасету MIRFlickr-20k. Було знайдено що всі канали охоплюють увесь спектр значень. Було знайдено мат. очікування – 111,5 для чевоного каналу, 103,5 для зеленого і 95,2 для синьогоканалу відповідно і дисперсію 4691,7, 4296,3 і 4171,9.

Було побудовано вектори даних та знайдено Гаусовські моделі для одновимірного та багатовимірних варіантів в залежності від кількості даних.

За допомогою методу головних компонент було відновлено тестові зображення та показано, що при збільшенні кількості компонент зростає якість відновлення. Також було помічено та проаналізовано помилки відновлення. Зібравши дані, було побуловано графік залежності середньої квадратичної похибки відновлених дображень від кількості компонент. Було помічено експоненціальну залежність, що свідчить про значні зміни при невеликих кількостях компонент (до 60) та майже непомітні при великих значеннях (більше 200).

Було проведено моделювання окремих каналів кольору зображень з використанням марковських ланцюгів, та сформовано стохастичні матриці за різними типами обходів. Також за даними було побудовано графічну модель марківського ланцюга. З моделі харківського ланцюга видно скупчення схожих яскравостей та не планий перехід від великих значень до малих. Це говорить про наявність різких зміщень в кольоровій гамі пікселів зображень.