МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. І.СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра фізико-технічних засобів захисту інформації

Лабораторна робота № 3

з дисципліни: «Автоматизація обробки ІзОД»

Керівник:
Прогонов Дмитро Олександрович

Виконав:
студент 5 курсу
групи ФЕ-91мп
Соколовський Владислав

Захищено з оцінкою

_____

_____

дата, підпис

Київ – 2020 р.

I. Вступ

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 повинно збігатися з номером студента в загальному списку групи) за модулем кількості зображень в тестовому пакеті.

Завдання:

1. Сформувати тестову вибірку зображень з вихідного пакета;
2. Для кожного каналу кольору кожного зображення з тестового пакета обчислити наступні характеристики:
    a. Математичне сподівання і дисперсію;
    b. Коефіцієнти асиметрії та ексцесу (нормалізований);
3. Використовуючи моделі SPAM і CC-PEV (http://dde.binghamton.edu/download/feature_extractors/), розрахувати вектора характеристик кожного каналу кольору кожного зображення;
4. Отримані параметри зображень упакувати в матрицю ознак (кожен рядок відповідає результатам для окремого каналу кольору тестового зображення, кожен стовпець - параметру зображення). Матриці ознак побудувати окремо для статистичних характеристик зображень (1-4 центральні моменти), а також моделей SPAM и CC-PEV;
5. Побудувати вектор міток класів зображень. Назва цільового класу для кожного студента визначається згідно з позицією студента у списку групи (див. Вкладений файл - наприклад, для першого студента в списку групи цільової клас "explore", для другого студента - мітка "sky", для третього студента - мітка "nikon" і т.д.);
6. псевдовипадковий чином розділити вихідний пакет зображень на 2 рівні частини (тестова і контрольна підвибірки). З використанням тестової підвибірки провести настройку наступних класифікаторів:
    a. Лінійна регресія;
    b. Робастна регресія;
    c. Логістична регресія;
    d. Метод опорных векторів (SVM);

7. Використовуючи налаштовані класифікатори з п. 6 провести обробки зображень з контрольного підпакету. Оцінити ймовірності правильної класифікації (0 і 1 класи), а також ймовірності помилок першого (помилкове спрацьовування) і другого (пропуск цілі) роду;

8. Повторити пп. 6-10 разів для отримання усередненої точності класифікації

II. Хід роботи

Роботу виконуватимемо мовою Python.Також в роботі будуть використані такі бібліотеки як:

- Os
- Matplotlib
- Numpy
- Scipy
- Pandas
- та інші

1. Формування тестової вибірки зображень з вихідного пакета

Для цього скористаємося функцією numpy.random() що обирає випадкові числа з переданого масиву за допомогою генератора Мерсена.

```
def create_index_list(file_list,count):
    print('\n>>Creating list of indexes')
    index_list = np.random.random_integers(0,len(file_list)-1,count)
    filtered_file_list=list()
    for index in tqdm(index_list):
        filtered_file_list.append(file_list[index])
    return index_list,filtered_file_list
```

Також задамо початкове значення варіанту за допомогою функції numpy.random.seed()

```
def config_random_generator(seed=14):
    print('>>Configure generator')
    np.random.seed(int(seed))
    generator_info = np.random.get_state()
    return
```

Після цього отриманий масив зображень буде знаходитись в loaded_images в виді двомірного масиву з трьома значеннями яскравості в кожній комірці.

```
files = lib.create_list_files()
index_list,files = lib.create_index_list(files,config['countImages'])
```

Тепер сформуємо матрицю для збору статистичних даних.

```
        def get_images_info(image_list):
            data = {}
            print('\n\n\n\n>>Analyzing images')
            for color_index,color_name in enumerate(RGB):
                print('\t\t>>Get info about '+str(color_name)+' color')
                data[color_name] = pd.DataFrame()
                for image_index,image_name in tqdm(enumerate(image_list)):
                    data[color_name] = pd.concat([data[color_name],
pd.DataFrame(pd.DataFrame(get_image_info(image_index,image_name,color_index),
index=[0, ]))], ignore_index=True)

data[color_name].to_csv(path_or_buf='./output/lab1/'+color_name+'.csv',index=False)
                print('\t\t>>Write data to ./output/lab1/'+color_name+'.csv\n\n')
```

2. Знаходження статистичних даних

a. Максимальна / мінімальне значення

Маючи вихідний масив з кількістю пікселів відповідної яскравості для знаходження максимального значення потрібно йти з кінця масиву до першого ненульового значення.

Його індекс і казатиме про наявність пікселів відповідної яскравості. Для мінімального потрібно проробити те саме, але з початку.

b. Математичне сподівання і дисперсія

Для знаходження скористаємось відповідними формулами:

$$D[X] = \sum_{i=1}^{n} p_i (x_i - M[X])^2,$$

$$M\left[g(X)\right] = \sum_{i=1}^{\infty} g(x_i) p_i,$$

Де x_i наше значення яскравості, а p_i – ймовірність її появи. p_i можна знайти як кількість пікселів даної яскравості поділену на всю кількість пікселів

c. Коефіцієнти асиметрії та ексцесу

Для пошуку медіани та інтерквартального розмаху скористаємось функціями макету scipy:

```
sp.stats.skew(a), # коэффициент асимметрии
sp.stats.kurtosis(a), # коэффициéнт эксцéсса
```

Після виконання коду:

```
def get_image_info(image_index,image_name,color_index):
    image = np.array(Image.open(image_name))
    a = image[ :, color_index].ravel()
    d = {
        'name': image_name,  # название файла
        'min': np.nanmin(a),  # минимум
        'max': np.nanmax(a),  # максимум
        'mean': np.nanmean(a),   # среднеарифметическое
        'var': np.nanvar(a), # дисперсия
        'median': np.nanmedian(a),  # медиана
        'average': np.average(a), # средневзвешенное(мат ожидание)
        'std': np.nanstd(a), # среднеквадратичное (стандартное) отклонение
        'skewness': sp.stats.skew(a), # коэффициент асимметрии
        'kurtosis': sp.stats.kurtosis(a), # коэффициéнт эксцéсса
        'interquartile range': sp.stats.iqr(a), # интерквартильный размах
        'best distribution': get_image_histogram(image_index,image_name)
        }
    return d
```

отримаємо наступні значення для кожного кольру кожного сображення:

```
average,best distribution,interquartile
range,kurtosis,max,mean,median,min,name,skewness,std,var
101.58758758758759,beta,60.0,-0.7437830509563512,172,101.58758758758759,110.0,9,./input/
mirflickr/im19052.jpg,-0.5088496333246547,39.04490479789601,1524.504590676763
246.2867540029112,beta,9.0,-0.8300230366027579,255,246.2867540029112,247.0,233,./input/m
irflickr/im13657.jpg,-0.2473624533422034,5.679390133401192,32.255472287374815
147.84733333333332,beta,4.0,-0.19336374177636895,155,147.84733333333332,148.0,137,./inpu
t/mirflickr/im9485.jpg,-0.3831508928596452,3.130925244857962,9.802692888888888
0.3774104683195592,beta,0.0,63.08458248026247,14,0.3774104683195592,0.0,0,./input/mirfli
ckr/im18839.jpg,6.621544503029918,1.0520801457914941,1.1068726331686514
177.44544544544544,laplace,97.0,-0.723479287578332,255,177.44544544544544,196.0,1,./inpu
t/mirflickr/im22856.jpg,-0.667994086672298,57.617693249141176,3319.798575352129
```

3. **Використовуючи моделі SPAM і CC-PEV (http://dde.binghamton.edu/download/feature_extractors/), розрахувати вектора характеристик кожного каналу кольору кожного зображення**

За допомогою моделі SPAM та CC-PEV в Matlabбуло отримано набір ознак зображень із тестової вибірки, дані заносимо до файлів, де кожному рядку відповідає зображення, для якого розраховувався набір ознак, а у кожному стовпчику значення ознак для одного зображення, щоб подальшому використати дані для формування матриці ознак та навчання класифікаторів.

4. **Отримані параметри зображень упакувати в матрицю ознак (кожен рядок відповідає результатам для окремого каналу кольору тестового зображення, кожен стовпець - параметру зображення). Матриці ознак побудувати окремо для статистичних характеристик зображень (1-4 центральні моменти), а також моделей SPAM и CC-PEV**

```
matrix = []
for sign in signs:
   m = np.array((RGB['red'][sign], RGB['green'][sign], RGB['blue'][sign]))
   matrix.append(m)
matrix = np.array(matrix)
output_file_path = './output/' + output_dir + '/matrix.csv'
with open(output_file_path, 'w') as f:
   csv.writer(f, delimiter=' ').writerows(matrix)

return name_list, data
```

```
[[[ 2.009220e+02  1.002400e+01  7.267500e+01 ...  1.228070e+02
    1.086020e+02  1.665500e+01]
  [ 2.016910e+02  9.835000e+00  7.276300e+01 ...  1.233540e+02
    1.099800e+02  1.708500e+01]
  [ 2.028410e+02  8.789000e+00  7.135700e+01 ...  1.249160e+02
    1.107910e+02  1.712000e+01]]

 [[ 1.185162e+03  1.032210e+02  4.348574e+03 ...  2.375245e+03
    2.353662e+03  3.617500e+01]
  [ 1.174480e+03  8.082800e+01  4.576116e+03 ...  2.657519e+03
```

```
    2.290743e+03  3.394400e+01]
  [ 1.170447e+03  5.818500e+01  4.612155e+03 ...  2.698558e+03
    2.302903e+03  3.477800e+01]]

 [[-7.310000e-01  2.637000e+00  1.030000e+00 ...  2.340000e-01
   -9.170000e-01  5.100000e-01]
  [-7.280000e-01  2.654000e+00  1.020000e+00 ...  2.010000e-01
   -9.590000e-01  4.340000e-01]
  [-7.620000e-01  2.322000e+00  1.060000e+00 ...  1.690000e-01
   -9.570000e-01  5.710000e-01]]

 [[-6.550000e-01  1.020200e+01 -6.100000e-02 ... -4.460000e-01
   -8.280000e-01  1.170000e-01]
  [-7.300000e-01  1.068800e+01 -1.330000e-01 ... -5.830000e-01
   -7.300000e-01  1.800000e-02]
  [-6.370000e-01  8.054000e+00 -2.300000e-02 ... -5.860000e-01
   -7.520000e-01  3.360000e-01]]]
```

5. **Побудувати вектор міток класів зображень. Назва цільового класу для кожного студента визначається згідно з позицією студента у списку групи (див. Вкладений файл - наприклад, для першого студента в списку групи цільової клас "explore", для другого студента - мітка "sky", для третього студента - мітка "nikon" і т.д.)**

Згідно варіанту наш цільовий клас - `'night'`

```
def get_tags(name_list):
   file_list_rand = create_list_files(target_format_in='.txt',
                                      path_in='./input/mirflickr/tags/')


   num_list = []
   for n in name_list:
       num_list.append(int((n.split('.')[-2]).split('m')[-1]))


   n = 0
   labels_list = []
   for file in file_list_rand:
           df1 = {
               'name': 0,
               'label': None,
           }
           nam = 'im' +
str(((file.split('/')[-1]).split('.')[-2]).split('s')[-1]) + '.jpg'
           try:
               tags = (open(file, 'r')).readlines()
```

```
            tags = [line.rstrip() for line in tags]
            if 'night' in tags:
                df1['name'] = nam
                df1['label'] = 0
                n += 1
            else:
                df1['name'] = nam
                df1['label'] = 1
            labels_list.append(df1)
        except:
            df1['name'] = nam
            df1['label'] = 1
            labels_list.append(df1)
            pass


return pd.DataFrame(labels_list), n
```

Отримуємо файл з мітками 0 або 1 для кожного зображення. Далі мержимо його з файлами з іншими даними, використовуючи в якості ключа номер зображення.

6. **Псевдовипадковий чином розділити вихідний пакет зображень на 2 рівні частини (тестова і контрольна підвибірки). З використанням тестової підвибірки провести настройку наступних класифікаторів:**
    a. **Лінійна регресія;**
    b. **Робастна регресія;**
    c. **Логістична регресія;**
    d. **Метод опорных векторів (SVM)**
7. **Використовуючи налаштовані класифікатори з п. 6 провести обробки зображень з контрольного підпакету. Оцінити ймовірності правильної класифікації (0 і 1 класи), а також ймовірності помилок першого (помилкове спрацьовування) і другого (пропуск цілі) роду**
8. **Повторити пп. 6-10 разів для отримання усередненої точності класифікації**

Тож формуємо песвдовипадковим чином тестову та контрольну вибірки зображень, розділивши дані на дві рівні частини, з використанням тестовової вибірки налаштовуємо наступні класификатори: LinearRegression(), LogisticRegression(), HuberRegressor(), SVC()
Запускаємо навчання 10 разів для статистичних характеристик зображень та даних моделей SPAM и CC-PEV, щоб отримати середню точність. В якості метрики оцінювання використовуємо стандартну accuracy score та суму помилок першого та другого роду

```
model_list = [LinearRegression(), LogisticRegression(),
HuberRegressor(max_iter=4000), SVC()]

def classifiers(data, n):

   df = pd.read_csv(data)
   true_label = df[(df.label ==
0)].sample(frac=1).reset_index(drop=True)[:n]
   false_label = df[(df.label ==
1)].sample(frac=1).reset_index(drop=True)[:n]
   DF = pd.concat([true_label, false_label])
   DF.to_csv('./output/'+output_dir+'/data.csv', index=False)
```

```
    df = pd.read_csv(data)
    X = df.iloc[:, 2:-1]
    y = df['label']
    for model in model_list:
        print(str(model))
        confusion_list = []
        accuracy_list = []
        for t in tqdm(range(10)):
           print(str(t+1)+' iteration')
           X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.5, random_state=42)
           model.fit(X_train, y_train)
           pred = model.predict(X_test)
           confusion_1 = 0
           y_test = np.array(y_test)
           for i in range(len(pred)):
               if pred[i] != y_test[i]:
                   confusion_1 += 1


           confusion_2 = 1
           for j in range(len(pred)):
               if y_test[i] == 0 and pred[i] != y_test[i]:
                   confusion_2 += 1
           confusion_list.append(confusion_1 + confusion_2)
           accuracy_list.append(accuracy_score(y_test, pred.round()))
        print('average error:')
        print(sum(confusion_list)/len(confusion_list))
        print('average accuracy:')
        print(sum(accuracy_list) / len(accuracy_list))
```

Отримано такі результати точності навчання класифікаторів на вибірці з статистичних характеристик зображень для червоного каналу кольору:

LinearRegression - 0.9920000000000002
LogisticRegression - 0.982780342445289
HuberRegressor - 0.96263749290374903
SVC - 0.970000000000002