BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN ĐHQG-HCM
KHOA TOÁN – TIN HỌC
---o0o---

BTTH TUẦN 2: KHAI THÁC DỮ LIỆU

Sinh viên thực hiện: Nguyễn Công Hoài Nam
Mã số sinh viên: 21280099

Tp. Hồ Chí Minh, ngày 6 tháng 4 năm 2024

1. Tính chuẩn p = (1, 2, ∞) cho 50 dòng đầu tiên của array mục 1

```
def calculate_p_norms(array, p):

    size = len(array)
    norms = np.empty((size,size))

    for i in range(size):
        for j in range(size):
            norms[i][j] = np.linalg.norm(array[i] - array[j],p)

    return norms

# Gọi hàm và lấy kết quả
first_50_rows = array[:50]
p_values = [1, 2, np.inf]
for p in p_values:
    norms = calculate_p_norms(first_50_rows, p)
    print(f"Chuẩn L_{p} cho 50 dòng đầu tiên :\n", norms)
    print()
```

Kết quả:

```
Chuẩn L_1 cho 50 dòng đầu tiên :
 [[ 0.      13.08095  5.35971 ... 15.98462  7.74504 21.06072]
 [13.08095  0.      15.80038 ... 17.53049 17.89603 18.96035]
 [ 5.35971 15.80038  0.      ... 15.88161  4.16871 19.98939]
 ...
 [15.98462 17.53049 15.88161 ...  0.      16.76954 19.93674]
 [ 7.74504 17.89603  4.16871 ... 16.76954  0.      21.72226]
 [21.06072 18.96035 19.98939 ... 19.93674 21.72226  0.     ]]

Chuẩn L_2 cho 50 dòng đầu tiên :
 [[0.         2.77635893 1.1697276  ... 3.74495828 1.59090253 4.40093478]
 [2.77635893 0.         3.38003304 ... 3.90492063 3.91433755 3.96620878]
 [1.1697276  3.38003304 0.         ... 3.82755203 0.97067099 4.35585992]
 ...
 [3.74495828 3.90492063 3.82755203 ... 0.         4.15300217 4.81090468]
 [1.59090253 3.91433755 0.97067099 ... 4.15300217 0.         4.89191224]
 [4.40093478 3.96620878 4.35585992 ... 4.81090468 4.89191224 0.        ]]

Chuẩn L_inf cho 50 dòng đầu tiên :
 [[0.      1.12221 0.45772 ... 1.85243 0.5728  1.60536]
 [1.12221 0.      1.10868 ... 1.93035 1.45054 1.63147]
 [0.45772 1.10868 0.      ... 2.      0.42872 1.54591]
 ...
 [1.85243 1.93035 2.      ... 0.      1.98103 2.     ]
 [0.5728  1.45054 0.42872 ... 1.98103 0.      1.93369]
 [1.60536 1.63147 1.54591 ... 2.      1.93369 0.     ]]
```

2. Tính láng giếng gần sử dụng độ đo tương đồng và độ đo tần suất xuất hiện ngược

a) Độ đo tương đồng

Ý tưởng: duyệt lần lượt từng phần tử của hai bản ghi `X` và `Y`, nếu `X[i] = Y[i]` thì cộng thêm một vào độ đo (nếu không bằng thì bằng 0)

```
def simple_similarity(X, Y):
    sim = 0
    for i in range(len(X)):
```

```
        if X[i] == Y[i]:
            sim += 1

    return sim /len(X)
```

b) Độ đo tần suất xuất hiện ngược

Ý tưởng:

- Muốn độ đo tần số xuất hiện ngược thì cần phải tính pk là một tỉ số của các bản ghi mà thuộc tính thứ k lấy giá trị x trong tập dữ liệu
- Có nghĩa là tính tỉ số của giá trị duy nhất x so với tổng giá trị của cột k

```
# Tính pk
def calculate_pk(array):
    pk_dict = {}
    num_rows, num_cols = array.shape

    for col_index in range(num_cols):
        col_values = array[:, col_index]
        unique_values, value_counts = np.unique(col_values,
return_counts=True)

        column_ratio = {}
        for value, count in zip(unique_values, value_counts):
            column_ratio[value] = count / num_rows

        pk_dict[col_index] = column_ratio

    return pk_dict
```

Sau đó tính tần số xuất hiện ngược (iof) bằng $\frac{1}{p_k(x)^2}$ nếu `X[i] = Y[i]` ngược lại bằng 0

```
# Độ đo tần suất xuất hiện ngược
# iof = inverse occurrence frequency
def iof_similarity(X, Y, pk_dict):
    sim = 0
    for i in range(len(X)):
        if X[i] == Y[i]:
            sim += 1 / pk_dict[i][X[i]] ** 2

    return sim /len(X)
```

c) Tìm láng giềng gần sử dụng hai phương pháp trên

Từ cách tính độ đo trên ta tính láng giềng gần bằng cách:

- Duyệt lần lượt mảng
- Lấy dòng có độ đo lớn nhất làm láng giềng gần

```
def find_nearest_neighbors(array, num_rows, similarity_measure):
    pk_dict = calculate_pk(array)
    neighbors = {}
    similarities = {}

    for i in range(num_rows):
        current_row = array[i]
        best_similarity = -1
        best_neighbor_index = -1

        for j in range(len(array)):
            if i == j:
                continue #Trùng hàng

            neighbor_row = array[j]

            if similarity_measure == 'simple':
                similarity = simple_similarity(current_row, neighbor_row)

              elif similarity_measure == 'iof':
                similarity = iof_similarity(current_row, neighbor_row,
pk_dict)

            else:
                raise ValueError("'simple' or 'iof' only")

            if similarity > best_similarity:
                best_similarity = similarity
                best_neighbor_index = j

        neighbors[i] = best_neighbor_index
        similarities[i] = best_similarity

    return neighbors , similarities
```

Vì bộ dữ liệu rất lớn nên để tránh mất thời gian em chỉ xét 100 dòng đầu của dữ liệu

Kết quả:

```
def show_result(array, nrows, measures):
    neighbors, similarities = find_nearest_neighbors(array, nrows, measures)

    df = pd.DataFrame({'Row': range(nrows), 'Nearest Neighbor':
neighbors.values(), 'Similarity Measure': similarities.values()})
    df = df.reset_index(drop=True)

    return df
```

Láng giếng gần bằng độ đo tương đồng

	Row	Nearest Neighbor	Similarity Measure
0	0	11	0.973684
1	1	2	0.973684
2	2	1	0.973684
3	3	1	0.973684
4	4	1	0.973684
...	...	...	...
95	95	85	1.000000
96	96	77	1.000000
97	97	77	1.000000
98	98	77	1.000000
99	99	77	1.000000

100 rows × 3 columns

Láng giếng gần bằng độ đo tần số xuất hiện ngược

	Row	Nearest Neighbor	Similarity Measure
0	0	11	1.439652
1	1	64	11.522535
2	2	65	11.522535
3	3	66	11.522535
4	4	67	11.522535
...	...	...	...
95	95	85	12.019998
96	96	77	1.991145
97	97	77	1.991145
98	98	77	1.991145
99	99	77	1.991145

100 rows × 3 columns