KHAI THÁC DỮ LIỆU

Nguyễn Xuân Việt Đức - 22280012

Bài tập thực hành - Lần 2

BÀI 2: SỰ TƯƠNG ĐỒNG VÀ CÁC KHOẢNG CÁCH

III. Nội dung thực hành:

1. Khoảng cách giữa các điểm trong dữ liệu số:

Viết hàm tính các chuẩn $p = 1, 2, \infty$ cho 50 dòng đầu tiên.

- **Chuẩn Manhattan** ($p = 1$): Đo khoảng cách giữa hai điểm trong không gian n-chiều bằng tổng độ lớn tuyệt đối của sự khác biệt giữa các tọa độ tương ứng. Đây còn gọi là "đường đi của taxi" vì nó tính khoảng cách theo các đoạn đường thẳng ngang và dọc.
- **Chuẩn Euclidean** ($p = 2$): Là khoảng cách "thẳng" giữa hai điểm trong không gian n-chiều, tính toán bằng cách lấy căn bậc hai của tổng các bình phương sự khác biệt giữa các tọa độ tương ứng. Đây là cách đo khoảng cách quen thuộc nhất và thường được dùng trong không gian Euclid.
- **Chuẩn Chebyshev** ($p = \infty$): Đo khoảng cách giữa hai điểm trong không gian n-chiều bằng độ lớn tuyệt đối lớn nhất của sự khác biệt giữa các tọa độ tương ứng. Khoảng cách này còn gọi là "khoảng cách vua" trong cờ vua, vì nó tính khoảng cách dựa trên số bước đi tối thiểu của vua (có thể đi theo mọi hướng).
- Với mỗi cặp điểm (i, j), tính khoảng cách giữa hai điểm này theo ba chuẩn:
  - Chuẩn Manhattan ($p = 1$): tổng giá trị tuyệt đối của sự chênh lệch giữa các tọa độ tương ứng.
  - Chuẩn Euclidean ($p = 2$): căn bậc hai của tổng các bình phương sự chênh lệch giữa các tọa độ tương ứng.
  - Chuẩn Chebyshev ($p = \infty$): giá trị tuyệt đối lớn nhất của sự chênh lệch giữa các tọa độ tương ứng.

```python
def calculate_norms(X, n_rows=50):

    array = X.values if isinstance(X, pd.DataFrame) else X
    array = array[:n_rows]

    row_pairs = []
    norm_1_distances = []
    norm_2_distances = []
    norm_inf_distances = []

    for i in range(len(array)):
        for j in range(i+1, len(array)):
            point_i = array[i, :]
            point_j = array[j, :]

            dist_p1 = np.linalg.norm(point_i - point_j, 1)      # p=1 (Manhattan)
            dist_p2 = np.linalg.norm(point_i - point_j)         # p=2 (Euclidean)
            dist_pinf = np.linalg.norm(point_i - point_j, np.inf)  # p=infinity (Chebyshev)

            row_pairs.append(f"({i}, {j})")
            norm_1_distances.append(dist_p1)
            norm_2_distances.append(dist_p2)
            norm_inf_distances.append(dist_pinf)

    results = pd.DataFrame({
        'Point Pair': row_pairs,
        'p=1 (Manhattan)': norm_1_distances,
        'p=2 (Euclidean)': norm_2_distances,
        'p=∞ (Chebyshev)': norm_inf_distances
    })

    return results
```

Output:

	Point Pair	p=1 (Manhattan)	p=2 (Euclidean)	p=∞ (Chebyshev)
0	(0, 1)	13.08095	2.776359	1.12221
1	(0, 2)	5.35971	1.169728	0.45772
2	(0, 3)	21.05729	4.772563	1.60536
3	(0, 4)	6.21387	1.377347	0.53525
4	(0, 5)	15.16631	3.049072	0.97202
...	...	...	...	...
1220	(46, 48)	2.74938	0.680880	0.26250
1221	(46, 49)	23.13494	5.038499	1.95288
1222	(47, 48)	16.76954	4.153002	1.98103
1223	(47, 49)	19.93674	4.810905	2.00000
1224	(48, 49)	21.72226	4.891912	1.93369

[1225 rows x 4 columns]

```
Manhattan distance matrix shape: (50, 50)
Euclidean distance matrix shape: (50, 50)
Chebyshev distance matrix shape: (50, 50)

Manhattan distance matrix (p=1):
[[ 0.   12.65  5.29 ... 15.11  7.53 20.61]
 [12.65  0.   15.44 ... 17.09 17.68 18.94]
 [ 5.29 15.44  0.   ... 15.08  4.03 19.61]
 ...
 [15.11 17.09 15.08 ...  0.   16.11 19.52]
 [ 7.53 17.68  4.03 ... 16.11  0.   21.48]
 [20.61 18.94 19.61 ... 19.52 21.48  0.  ]]

Euclidean distance matrix (p=2):
[[0.   2.74 1.17 ... 3.64 1.58 4.38]
 [2.74 0.   3.36 ... 3.88 3.91 3.97]
 [1.17 3.36 0.   ... 3.74 0.96 4.34]
 ...
 [3.64 3.88 3.74 ... 0.   4.1  4.79]
 [1.58 3.91 0.96 ... 4.1  0.   4.89]
 [4.38 3.97 4.34 ... 4.79 4.89 0.  ]]

Chebyshev distance matrix (p=∞):
[[0.   1.12 0.46 ... 1.85 0.57 1.61]
 [1.12 0.   1.11 ... 1.93 1.45 1.63]
 [0.46 1.11 0.   ... 2.   0.43 1.55]
 ...
 [1.85 1.93 2.   ... 0.   1.98 2.  ]
 [0.57 1.45 0.43 ... 1.98 0.   1.93]
 [1.61 1.63 1.55 ... 2.   1.93 0.  ]]
```

Nhận xét tổng thể:

- Manhattan tạo ra giá trị cao nhất trong số ba khoảng cách do cách tính tổng khoảng cách trên từng trục.
- Euclidean tạo ra giá trị trung bình do tính theo đường thẳng giữa hai điểm.
- Chebyshev cho giá trị nhỏ nhất, phù hợp với các bài toán mà chỉ quan tâm đến bước nhảy lớn nhất giữa các trục.

2. Độ đo tương đồng giữa các điểm trong tập dữ liệu phân loại

Tính các láng giềng gần nhất ở mục 2 sử dụng mục 2 với độ đo Overlap và độ đo tần suất xuất hiện ngược.

Độ đo Overlap: Độ đo Overlap giữa hai điểm dữ liệu phân loại được tính bằng số lượng thuộc tính có giá trị khác nhau. Độ đo này rất hữu ích trong việc so sánh sự tương đồng giữa các điểm dữ liệu phân loại.

- Công thức: $D_{overlap}(x, y) = \sum_{i=1}^{n} \mathbb{I}(x_i \neq y_i)$, với $\mathbb{I}$ là hàm chỉ thị.

Độ đo tần suất xuất hiện ngược: Độ đo tần suất xuất hiện ngược giữa hai điểm dữ liệu được tính bằng cách sử dụng tần suất xuất hiện của các giá trị. Các giá trị ít xuất hiện sẽ có trọng số lớn hơn, do đó sự khác biệt về các giá trị hiếm sẽ được nhấn mạnh hơn so với các giá trị phổ biến.

- Công thức: $D_{if}(x, y) = \sum_{i=1}^{n} \mathbb{I}(x_i \neq y_i) \left(\frac{1}{freq(x_i)} + \frac{1}{freq(y_i)} \right) /2$

Các bước thực hiện:

1. Tính tần suất xuất hiện của từng giá trị trong mỗi cột của tập dữ liệu.
2. Tính khoảng cách giữa các điểm dữ liệu sử dụng độ đo Overlap và độ đo tần suất xuất hiện ngược.
3. Tìm các láng giềng gần nhất cho mỗi điểm dữ liệu dựa trên các khoảng cách đã tính.

Output:

```
Overlap Distance Matrix (first 10x10):
[[0. 1. 1. 1. 2. 1. 1. 1. 2. 2.]
 [1. 0. 1. 2. 3. 2. 2. 2. 3. 3.]
 [1. 1. 0. 2. 3. 2. 2. 2. 3. 3.]
 [1. 2. 2. 0. 2. 1. 1. 1. 2. 2.]
 [2. 3. 3. 2. 0. 2. 2. 2. 2. 3.]
 [1. 2. 2. 1. 2. 0. 1. 1. 2. 2.]
 [1. 2. 2. 1. 2. 1. 0. 1. 2. 2.]
 [1. 2. 2. 1. 2. 1. 1. 0. 2. 1.]
 [2. 3. 3. 2. 2. 2. 2. 2. 0. 3.]
 [2. 3. 3. 2. 3. 2. 2. 1. 3. 0.]]
```

Hình 1: Ma trận khoảng cách Overlap

```
Inverse Frequency Distance Matrix (first 10x10):
[[0.         0.14572536 0.19617389 0.12862497 0.8564846  0.08271983
  0.10589181 0.1691263  0.88285491 0.31137983]
 [0.14572536 0.         0.06319726 0.27435033 1.00220996 0.22844519
  0.25161717 0.31485166 1.02858027 0.45710519]
 [0.19617389 0.06319726 0.         0.32479886 1.05265849 0.27889372
  0.3020657  0.36530019 1.0790288  0.50755372]
 [0.12862497 0.27435033 0.32479886 0.         0.84541733 0.07165256
  0.09203537 0.14836841 0.87178764 0.29062194]
 [0.8564846  1.00220996 1.05265849 0.84541733 0.         0.81879441
  0.8320796  0.86995965 0.13195058 1.01221317]
 [0.08271983 0.22844519 0.27889372 0.07165256 0.81879441 0.
  0.05831483 0.09619488 0.84516472 0.2384484 ]
 [0.10589181 0.25161717 0.3020657  0.09203537 0.8320796  0.05831483
  0.         0.1226369  0.85844991 0.26489043]
 [0.1691263  0.31485166 0.36530019 0.14836841 0.86995965 0.09619488
  0.1226369  0.         0.89632995 0.14225353]
 [0.88285491 1.02858027 1.0790288  0.87178764 0.13195058 0.84516472
  0.85844991 0.89632995 0.         1.03858348]
 [0.31137983 0.45710519 0.50755372 0.29062194 1.01221317 0.2384484
  0.26489043 0.14225353 1.03858348 0.        ]]
```

Hình 2: Ma trận khoảng cách tần suất xuất hiện ngược

Nhận xét tổng thể:

- Độ đo Overlap đơn giản và dễ hiểu, phù hợp với các bài toán phân loại cơ bản.
- Độ đo tần suất xuất hiện ngược phức tạp hơn nhưng hữu ích trong các bài toán mà sự khác biệt của các giá trị hiếm cần được nhấn mạnh.