# KHAI THÁC DỮ LIỆU

Nguyễn Xuân Việt Đức - 22280012

Bài tập thực hành - Lần 4

**Bài 4: KHAI THÁC MẪU KẾT HỢP**

**II. Nội dung thực hành:**

**1. Cài đặt thuật toán Apriori**

*Đọc dữ liệu từ file csv:*

```
# Load dữ liệu
df = pd.read_csv('dataW4.csv', header=None)
print(df)
```

```
       0      1       2       3     4       5
0   Wine  Chips  Bread   Butter  Milk  Apple
1   Wine    NaN  Bread   Butter  Milk    NaN
2    NaN    NaN  Bread   Butter  Milk    NaN
3    NaN  Chips     NaN     NaN   NaN  Apple
4   Wine  Chips  Bread   Butter  Milk  Apple
5   Wine  Chips     NaN     NaN  Milk    NaN
6   Wine  Chips  Bread   Butter   NaN  Apple
7   Wine  Chips     NaN     NaN  Milk    NaN
8   Wine    NaN  Bread      NaN   NaN  Apple
9   Wine    NaN  Bread   Butter  Milk    NaN
10   NaN  Chips  Bread   Butter   NaN  Apple
11  Wine    NaN     NaN  Butter  Milk  Apple
12  Wine  Chips  Bread   Butter  Milk    NaN
13  Wine    NaN  Bread      NaN  Milk  Apple
14  Wine    NaN  Bread   Butter  Milk  Apple
15  Wine  Chips  Bread   Butter  Milk  Apple
16   NaN  Chips  Bread   Butter  Milk  Apple
17   NaN  Chips     NaN  Butter  Milk  Apple
18  Wine  Chips  Bread   Butter  Milk  Apple
```

Chuyển DataFrame thành dạng danh sách (list)

```
data = df.copy()

records = []
for i in range(0, data.shape[0]):
    records.append([str(data.values[i,j]) for j in range(0, data.shape[1])])
```

Chuyển records thành dạng transaction

```
te = TransactionEncoder()
te_ary = te.fit(records).transform(records)
df1 = pd.DataFrame(te_ary, columns=te.columns_)
print(df1)
```

| | Apple | Bread | Butter | Chips | Milk | Wine | nan |
|---|---|---|---|---|---|---|---|
| 0 | True | True | True | True | True | True | False |
| 1 | False | True | True | False | True | True | True |
| 2 | False | True | True | False | True | False | True |
| 3 | True | False | False | True | False | False | True |
| 4 | True | True | True | True | True | True | False |
| 5 | False | False | False | True | True | True | True |
| 6 | True | True | True | True | False | True | True |
| 7 | False | False | False | True | True | True | True |
| 8 | True | True | False | False | False | True | True |
| 9 | False | True | True | False | True | True | True |
| 10 | True | True | True | True | False | False | True |
| 11 | True | False | True | False | True | True | True |
| 12 | False | True | True | True | True | True | True |
| 13 | True | True | False | False | True | True | True |
| 14 | True | True | True | False | True | True | True |
| 15 | True | True | True | True | True | True | False |
| 16 | True | True | True | True | True | False | True |
| 17 | True | False | True | True | True | False | True |
| 18 | True | True | True | True | True | True | False |
| 19 | True | True | True | False | True | True | True |
| 20 | True | True | False | True | True | True | True |
| 21 | False | False | False | True | False | False | True |

Xây dựng mô hình Apriori

```
frequent_itemsets = apriori(df1, min_support=0.6, use_colnames=True)
print(frequent_itemsets)
```

```
    support      itemsets
0  0.681818       (Apple)
1  0.727273      (Bread )
2  0.681818      (Butter)
3  0.636364       (Chips)
4  0.772727        (Milk)
5  0.727273        (Wine)
6  0.818182         (nan)
7  0.636364  (Wine, Milk)
```

Xây dựng luật kết hợp dựa trên ma trận hỗ trợ

```
rules = association_rules(frequent_itemsets, metric='support', support_only=True, min_threshold=0.1)
rules = rules[['antecedents', 'consequents', 'support']]
print(rules)
```

```
  antecedents consequents   support
0      (Wine)      (Milk)  0.636364
1      (Milk)      (Wine)  0.636364
```

## 2. Code lại thuật toán Apriori:

2.1. Giới thiệu về thuật toán Apriori:

Thuật toán Apriori là một thuật toán nền tảng trong khai thác tập phổ biến (frequent itemset mining) và tạo luật kết hợp (association rule mining). Thuật toán dựa trên nguyên tắc "nếu một tập hợp con không phổ biến, thì bất kỳ tập hợp cha nào chứa nó cũng không phổ biến" (nguyên tắc Apriori).

2.2. Cấu trúc lớp AprioriAlgorithm

Khởi tạo:

```
def __init__(self, min_support=0.5, min_confidence=0.7):
    self.min_support = min_support
    self.min_confidence = min_confidence
    self.itemsets = {}
    self.rules = []
    self.item_counts = {}
    self.itemset_counts = {}
```

Các tham số:

- `min_support`: Ngưỡng hỗ trợ tối thiểu (mặc định là 0.5)
- `min_confidence`: Ngưỡng tin cậy tối thiểu (mặc định là 0.7)
- `itemsets`: Từ điển lưu trữ các tập phổ biến theo kích thước
- `rules`: Danh sách lưu trữ các luật kết hợp
- `item_counts`: Đếm số lần xuất hiện của mỗi item
- `itemset_counts`: Đếm số lần xuất hiện của mỗi itemset

Phương thức huấn luyện (fit):

- Bước 1: Tìm các tập phổ biến gồm 1 phần tử (`_find_frequent_1_itemsets`)
- Bước 2: Lặp qua các giá trị k (kích thước tập phổ biến) từ 2 trở đi:
  - Tạo các tập ứng viên có kích thước k (`_generate_candidate_itemsets`)
  - Đếm tần suất xuất hiện của các tập ứng viên (`_count_itemsets`)
  - Lọc ra các tập phổ biến dựa trên ngưỡng `min_support`
  - Tăng k và tiếp tục cho đến khi không còn tập phổ biến mới
- Bước 3: Tạo các luật kết hợp từ các tập phổ biến (`_generate_rules`)

Tìm tập phổ biến 1 phần tử:

- Phương thức `_find_frequent_1_itemsets` đếm số lần xuất hiện của mỗi item trong tập dữ liệu giao dịch
- Lọc ra các item có độ hỗ trợ (support) lớn hơn hoặc bằng ngưỡng `min_support`
- Kết quả được lưu trong `self.itemsets[1]`

Tạo tập ứng viên:

- Phương thức `_generate_candidate_itemsets(k)` tạo các tập ứng viên có kích thước k bằng cách kết hợp các tập phổ biến có kích thước k-1
- **Kết hợp**: Ghép cặp các tập phổ biến có kích thước k-1 để tạo ra các tập ứng viên có kích thước k
- **Loại bỏ**: Loại bỏ các tập ứng viên có tập con không phổ biến dựa trên nguyên tắc Apriori

Kiểm tra tập con phổ biến:

- Phương thức `_all_subsets_frequent` kiểm tra xem tất cả các tập con của một tập ứng viên có phải là tập phổ biến hay không
- Dựa trên nguyên tắc Apriori: nếu một tập con không phổ biến, thì tập ứng viên cũng không phổ biến
- Giúp giảm số lượng tập ứng viên cần đếm

Đếm tần suất của tập ứng viên:

- Phương thức `_count_itemsets` đếm số lần xuất hiện của mỗi tập ứng viên trong tập dữ liệu giao dịch
- Một tập ứng viên được xem là xuất hiện trong một giao dịch nếu tất cả các phần tử của tập đó đều xuất hiện trong giao dịch
- Kết quả được lưu trong `self.itemset_counts[k]`

Tạo luật kết hợp:

- Phương thức `_generate_rules` tạo các luật kết hợp từ các tập phổ biến
- Cho mỗi tập phổ biến có kích thước từ 2 trở lên, thuật toán tạo ra các luật kết hợp theo dạng `antecedent → consequent`
- Độ tin cậy (confidence) được tính bằng công thức:

$$confidence(A \rightarrow B) = \frac{support(A \cup B)}{support(A)} \tag{1}$$

- Chỉ các luật có độ tin cậy lớn hơn hoặc bằng ngưỡng `min_confidence` mới được lưu lại

Phương thức hiển thị kết quả:

- `print_frequent_1_itemsets`: Hiển thị các tập phổ biến có 1 phần tử
- `print_candidate_itemsets`: Hiển thị các tập ứng viên kích thước k
- `print_frequent_k_itemsets`: Hiển thị các tập phổ biến kích thước k
- `print_rules`: Hiển thị các luật kết hợp đã tạo
- `get_itemsets`: Trả về tất cả các tập phổ biến
- `get_rules`: Trả về tất cả các luật kết hợp

2.3. Ưu điểm và nhược điểm của thuật toán Apriori

Ưu điểm:

- Dễ hiểu và triển khai
- Tận dụng tính chất Apriori để giảm không gian tìm kiếm
- Phù hợp với các tập dữ liệu nhỏ và trung bình
- Tạo ra các luật kết hợp rõ ràng và dễ diễn giải

Nhược điểm:

- Kém hiệu quả khi số lượng giao dịch và số lượng mặt hàng lớn
- Cần nhiều lần quét qua tập dữ liệu, đặc biệt là khi tạo ra các tập ứng viên lớn
- Sinh ra quá nhiều tập ứng viên mà sau đó bị loại bỏ
- Không hiệu quả khi làm việc với tập dữ liệu thưa (sparse datasets)

```
Loaded 21 transactions from dataW4.csv
Sample of transactions:
Transaction 1: ['Wine', 'Bread ', 'Butter', 'Milk']
Transaction 2: ['Bread ', 'Butter', 'Milk']
Transaction 3: ['Chips', 'Apple']
Transaction 4: ['Wine', 'Chips', 'Bread ', 'Butter', 'Milk', 'Apple']
Transaction 5: ['Wine', 'Chips', 'Milk']
```

```
Running Apriori with min_support=60% and min_confidence=80%

- Với k = 1, ta khởi tạo F1
Item            frequency       support
Apple           14              67%
Bread           15              71%
Butter          14              67%
Chips           13              62%
Milk            16              76%
Wine            15              71%
```

```
- Khởi tạo C2 bằng việc kết hợp các cặp item của F1
['Milk', 'Apple'], ['Butter', 'Milk'], ['Butter', 'Chips'], ['Wine', 'Chips'], ['Bread ', 'Milk'], ['Bread ', 'Chips'], ['Wine', 'Milk'], ['Bread ', 'Apple'],
['Chips', 'Apple'], ['Wine', 'Apple'], ['Butter', 'Bread '], ['Butter', 'Apple'], ['Chips', 'Milk'], ['Bread ', 'Wine'], ['Butter', 'Wine']
```

- Tạo F2

| Item | frequency | support |
|---|---|---|
| Apple, Bread | 11 | 52% |
| Apple, Butter | 10 | 48% |
| Apple, Chips | 9 | 43% |
| Apple, Milk | 10 | 48% |
| Apple, Wine | 10 | 48% |
| Bread , Butter | 12 | 57% |
| Bread , Chips | 8 | 38% |
| Bread , Milk | 12 | 57% |
| Bread , Wine | 12 | 57% |
| Butter, Chips | 8 | 38% |
| Butter, Milk | 12 | 57% |
| Butter, Wine | 10 | 48% |
| Chips, Milk | 9 | 43% |
| Chips, Wine | 8 | 38% |
| Milk, Wine | 13 | 62% |

- Tìm các hạng mục quan trọng dựa vào minsup = 60% nên ta lấy các 2-item sau:
[['Wine', 'Milk']]

- Phát sinh các luật
Wine → Milk có conf(Wine → Milk) = support(Wine, Milk)/support(Wine) = 87%
Milk → Wine có conf(Milk → Wine) = support(Milk, Wine)/support(Milk) = 81%

## 3. KẾT QUẢ SO SÁNH

- Thời gian thực thi:
  - Implementation tự viết: 0.0000 giây
  - Thư viện mlxtend: 0.0070 giây

- Số lượng itemsets:
  - Implementation tự viết: 7
  - Thư viện mlxtend: 7

- Số lượng luật:
  - Implementation tự viết: 2
  - Thư viện mlxtend: 2

- Kết quả có trùng khớp không:
  - Itemsets: Có
  - Luật: Có

## 4. MẪU CÁC ITEMSETS TÌM ĐƯỢC

- Implementation tự viết (top 5):
  1. ['Milk'] (support: 0.7619)
  2. ['Wine'] (support: 0.7143)
  3. ['Bread '] (support: 0.7143)
  4. ['Butter'] (support: 0.6667)
  5. ['Apple'] (support: 0.6667)

- Thư viện mlxtend (top 5):
  1. ['Milk'] (support: 0.7619)
  2. ['Bread '] (support: 0.7143)
  3. ['Wine'] (support: 0.7143)
  4. ['Apple'] (support: 0.6667)
  5. ['Butter'] (support: 0.6667)

5. MẪU CÁC LUẬT KẾT HỢP TÌM ĐƯỢC
- Implementation tự viết (top 5):
  1. ['Wine'] → ['Milk'] (support: 0.6190, confidence: 0.8667)
  2. ['Milk'] → ['Wine'] (support: 0.6190, confidence: 0.8125)

- Thư viện mlxtend (top 5):
  1. ['Wine'] → ['Milk'] (support: 0.6190, confidence: 0.8667)
  2. ['Milk'] → ['Wine'] (support: 0.6190, confidence: 0.8125)

6. KẾT LUẬN
- Implementation tự viết cho kết quả trùng khớp với thư viện mlxtend
- Thời gian thực thi quá nhanh để so sánh chính xác

=== Summary of Results ===
Total transactions processed: 21
Total frequent itemsets found: 7
Total association rules found: 2