ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - TIN HỌC

BTTH TUẦN 5: KHAI THÁC DỮ LIỆU

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

Ngày 11 tháng 5 năm 2024

MỤC LỤC

I. Sử dụng thư viện

- Cài đặt thư viện pyECLAT (đã cài đặt)
- Sử dụng lại file data.csv

```
import numpy as np
import pandas as pd
from pyECLAT import ECLAT


dataframe = pd.read_csv('data.csv', header = None)
dataframe
```

	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
19	Wine	NaN	Bread	Butter	Milk	Apple
20	Wine	Chips	Bread	NaN	Milk	Apple
21	NaN	Chips	NaN	NaN	NaN	NaN

Chuyển dữ liệu thành lớp ECLAT và khởi tạo DataFrame nhị phân

```
eclat_instance = ECLAT(data = dataframe, verbose = True)
eclat_instance.df_bin
```

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

Khởi tạo các luật kết hợp

```
# count items in each row
items_per_transaction = eclat_instance.df_bin.astype(int).sum(axis=1)
# the item should appear at least at 5% of transactions
min_support = 0.5
# start from transactions containing at least 2 items
min_combination = 2
# up to maximum items per transactions
max_combination = max(items_per_transaction)
rule_indices, rule_supports = eclat_instance.fit(min_support = min_support,
                                                 min_combination = min_combination,
                                                 max_combination = max_combination,
                                                 separator  = ' & ',
                                                 verbose = True)
result = pd.DataFrame(rule_supports.items(), columns=['Item','Support'])
# result = result.sort_values(by=['Support','Item'], ascending = False)
result
```

	Item	Support
0	Butter & Wine	0.500000
1	Butter & Milk	0.590909
2	Butter & Apple	0.500000
3	Butter & Bread	0.590909
4	Wine & Milk	0.636364
5	Wine & Apple	0.500000
6	Wine & Bread	0.590909
7	Milk & Apple	0.500000
8	Milk & Bread	0.590909
9	Apple & Bread	0.545455
10	Butter & Milk & Bread	0.500000
11	Wine & Milk & Bread	0.500000

II. Cài đặt lại thuật toán Vertical Apriori

1. Xử lý dữ liệu

Chuyển DataFrame về dạng List

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

Sau đó loại bỏ chuỗi 'nan' (không phải là một item trong itemsets)

```
records = [[item for item in i if item != 'nan'] for i in records]
records
```

```
[['Wine', 'Chips', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['Wine', 'Bread', 'Butter', 'Milk'],
 ['Bread', 'Butter', 'Milk'],
 ['Chips', 'Apple'],
 ['Wine', 'Chips', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['Wine', 'Chips', 'Milk'],
 ['Wine', 'Chips', 'Bread', 'Butter', 'Apple'],
 ['Wine', 'Chips', 'Milk'],
 ['Wine', 'Bread', 'Apple'],
 ['Wine', 'Bread', 'Butter', 'Milk'],
 ['Chips', 'Bread', 'Butter', 'Apple'],
 ['Wine', 'Butter', 'Milk', 'Apple'],
 ['Wine', 'Chips', 'Bread', 'Butter', 'Milk'],
 ['Wine', 'Bread', 'Milk', 'Apple'],
 ['Wine', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['Wine', 'Chips', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['Chips', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['Chips', 'Butter', 'Milk', 'Apple'],
 ['Wine', 'Chips', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['Wine', 'Bread', 'Butter', 'Milk', 'Apple'],
 ['Wine', 'Chips', 'Bread', 'Milk', 'Apple'],
 ['Chips']]
```

Hàm **convert_vertical** chuyển dataset từ dạng ngang (horizontal) sang dạng thẳng đứng (vertical)

```
def convert_vertical(records):
    transactions = {}
    for transaction_index, transaction in enumerate(records):
        for t in transaction:
```

```
            # frozenset là tập không thay đổi (immutable set) tức nó không cho phép chỉnh sửa
                , thay đổi hay giá trị lặp
            t = frozenset([t])
            if t not in transactions:
                transactions[t] = []
            transactions[t].append(transaction_index)
    return transactions
```

Kiểm tra kết quả

```
X = convert_vertical(records)
for i in X:
    print(i,"\t:",X[i])
```

```
frozenset({'Wine'})     : [0, 1, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 18, 19, 20]
frozenset({'Chips'})    : [0, 3, 4, 5, 6, 7, 10, 12, 15, 16, 17, 18, 20, 21]
frozenset({'Bread'})    : [0, 1, 2, 4, 6, 8, 9, 10, 12, 13, 14, 15, 16, 18, 19, 20]
frozenset({'Butter'})   : [0, 1, 2, 4, 6, 9, 10, 11, 12, 14, 15, 16, 17, 18, 19]
frozenset({'Milk'})     : [0, 1, 2, 4, 5, 7, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
frozenset({'Apple'})    : [0, 3, 4, 6, 8, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20]
```

2. Cài đặt thuật toán

Tóm tắt lý thuyết

- $\mathbf{C_k}$: k-item candidate itemset (tập ứng viên có k-hạng mục)
- $\mathbf{F_k}$: k-item frequent itemset (được lọc ra từ C_k với các giá trị thoã mãn '**minimum_support**')

Để áp dụng thuật toán *Apriori* ta cần khởi tạo trước tập C_1 từ đó tạo tập F_1
Hàm **`create_candidate_1_itemsets(X)`** tạo tập C_1 từ data X

```
# Vì C_1 chỉ có một item duy nhất nên ta chỉ cần lấy các key của database X
# Vì X là database dạng vertical nên các key của nó chính là các itemsets (tập hạng mục)
def create_candidate_1_itemsets(X):
    c1 = []
    for transaction in X:
        c1.append(transaction)
    return c1
```

Hàm **`find_num_transactions(X)`** tính tập F_K
Khác với *Horizontal Apriori*, thuật toán *Vertical Apriori* tính support sử dụng chiều dài của danh sách tid của itemset (nếu itemset có nhiều hơn một hạng mục thì danh sách tid chính là tập giao của các item đơn lẻ trong itemset)

```
# Hàm trả về tổng số giao dịch (với dữ liệu ngang thì nó là số dòng)
# Vì đây là dữ liệu dọc nên nó sẽ là số giao dịch lớn nhất trong danh sách tid
def find_num_transactions(X):

    num_transactions = 0
    for item in X:
        for tid in X[item]:
            if tid > num_transactions:
                num_transactions = tid

    # Trả về tổng số giao dịch + 1 (vì Python bắt đầu đếm từ 0)
    return num_transactions + 1

def create_frequent_itemsets(X, ck, min_support):
```

```
    # Danh sách tập hợp mục thường xuyên
    freq_item = []
    # Tạo từ điển để lưu trữ hỗ trợ của mỗi tập hợp mục
    item_support = {}
    # Tìm tổng số giao dịch trong tập dữ liệu
    num_transactions = find_num_transactions(X)

    # Duyệt qua từng tập hạng mục ứng viên (tập ck)
    for items in ck:
        # tid_list lưu trữ các tid chung của các tập hạng mục
        tid_list = None
        # Duyệt qua mỗi mục trong tập hạng mục
        for item in items:
            # Chuyển về dạng frozenset đảm bảo tính duy nhất và độc lập
            item = frozenset([item])
            # Nếu tid_list = None (tức là hạng mục đầu tiên)
            if tid_list is None:
                # Khởi tạo set với các hạng mục đầu tiên
                tid_list = set(X[item])
            else:
                # Thực hiện phép giao lấy tid chung của các hạng mục trong tập hạng mục
                tid_list = tid_list.intersection(X[item])
        # Tính support = số tid chung / tổng giao dịch
        item_support[items] = len(tid_list) / num_transactions

    # Duyệt qua từng tập hạng mục và kiểm tra support
    for item in item_support:
        # Nếu hỗ trợ lớn hơn hoặc bằng min_support, thêm vào danh sách tập hạng mục thường
            xuyên
        if item_support[item] >= min_support:
            freq_item.append(item)

    # Trả về danh sách tập hạng mục thường xuyên và từ điển hỗ trợ
    return freq_item, item_support
```

Hàm **create_candidate_k_itemsets(freq_item,k** tính C_k từ F_{k-1}

- Nếu k $= 1$, hàm tính C_2
- Nếu k $\neq 1$, hàm tính C_i với $i \neq 1$ (C_1 đã được tính từ trước)

```
# Hàm hỗ trợ tính tổ hợp từ tập hợp các phần tử cho trước
from itertools import combinations

# Hàm tạo k-item candidate itemsets
def create_candidate_k_itemsets(freq_item, k):
    ck = []

    # Với k = 1, tạo 2-item candidate itemsets (c2)
    if k == 1:
        for f1, f2 in combinations(freq_item, 2):
            item = f1 | f2 # union of two sets
            ck.append(item)
    # Với k != 1
    else:
        for f1, f2 in combinations(freq_item, 2):

            # Nếu k-itemset có k - 1 phần tử chung
            # sẽ lấy phần hợp của chúng vào k+1-itemset
            intersection = f1 & f2
            if len(intersection) == k - 1:
```

```
                item = f1 | f2
                if item not in ck:
                    ck.append(item)
    return ck
```

Từ các hàm con trên, ta có hàm **apriori_from_scratch(X, min_support, min_combination=1, max_combination=None)**:

- **X** là là tập dữ liệu chứa các giao dịch (vertical)
- **min_support** là ngưỡng hỗ trợ tối thiểu để một tập hạng mục được coi là thường xuyên.
- **max_combination** chỉ trả về các giao dịch có tối đa **max_combination** (mặc định bằng tất cả số hạng mục mà database có).

```
def apriori_from_scratch(X, min_support, min_combination=1, max_combination=None):

    # Khởi tạo max_combination nếu không được cung cấp
    if max_combination is None:
        max_combination = len(X)

    # Tạo tập hợp ứng viên 1-item (c1)
    c1 = create_candidate_1_itemsets(X)
    # Tạo tập hợp itemsets thường xuyên f1 từ c1
    freq_item, item_support_dict = create_frequent_itemsets(X, c1, min_support)
    # Thêm f1 vào freq_items (tập hợp các itemsets thường xuyên)
    freq_items = [freq_item]

    # Bắt đầu lặp từ k = 1
    k = 1
    while True:
        # Lấy tập hợp itemsets thường xuyên fk từ freq_items (được lưu dưới dạng list, bắt
            đầu từ index 0)
        freq_item = freq_items[k-1]
        # Tính tập hợp ứng viên ck+1 từ fk
        ck = create_candidate_k_itemsets(freq_item, k)
        # Tính tập hợp itemsets thường xuyên fk+1 từ ck+1 và từ điển hỗ trợ
        freq_item, item_support = create_frequent_itemsets(X, ck, min_support)
        # Nếu fk+1 trống => dừng
        # Kiểm tra điều kiện dừng để tránh thêm list rỗng vào freq_items
        if len(freq_item) == 0:
            break

        # Thêm fk+1 vào freq_items
        freq_items.append(freq_item)
        # Cập nhật từ điển hỗ trợ tổng
        item_support_dict.update(item_support)
        # Tăng k để tiếp tục lặp
        k += 1

    # Chọn các tập hợp itemsets thường xuyên trong khoảng min_combination đến max_combination
    freq_items = freq_items[min_combination-1:max_combination-1]

    return freq_items, item_support_dict
```

3. So sánh kết quả

Khởi tạo một số tham số (giống với thông số sử dụng hàm thư viện)

```
X = convert_vertical(records)
min_support = 0.5
```

```
min_combination = 2
max_combination = len(X)
```

Tính kết quả

```
freq_items, item_support_dict = apriori_from_scratch(X,min_support, min_combination,
    max_combination)
freq_items
```

```
[[frozenset({'Bread', 'Wine'}),
  frozenset({'Butter', 'Wine'}),
  frozenset({'Milk', 'Wine'}),
  frozenset({'Apple', 'Wine'}),
  frozenset({'Bread', 'Butter'}),
  frozenset({'Bread', 'Milk'}),
  frozenset({'Apple', 'Bread'}),
  frozenset({'Butter', 'Milk'}),
  frozenset({'Apple', 'Butter'}),
  frozenset({'Apple', 'Milk'})],
 [frozenset({'Bread', 'Milk', 'Wine'}),
  frozenset({'Bread', 'Butter', 'Milk'})]]
```

Để nhìn giống kết quả hàm thư viện ta format lại kết quả

```
def tidy(freq_items, item_support_dict):
    supports = []
    itemsets = []
    for itemset in freq_items:
        for item in itemset:
            supports.append(item_support_dict[item])
            itemsets.append(item)
    df = pd.DataFrame({'Itemset': itemsets, 'Support': supports})
    df_sorted = df.sort_values(by=['Support'], ascending=False)
    return df_sorted

tidy(freq_items, item_support_dict)
```

	Itemset	Support
2	(Wine, Milk)	0.636364
0	(Bread, Wine)	0.590909
4	(Bread, Butter)	0.590909
5	(Bread, Milk)	0.590909
7	(Butter, Milk)	0.590909
6	(Bread, Apple)	0.545455
1	(Butter, Wine)	0.500000
3	(Apple, Wine)	0.500000
8	(Apple, Butter)	0.500000
9	(Apple, Milk)	0.500000
10	(Bread, Wine, Milk)	0.500000
11	(Bread, Butter, Milk)	0.500000

Tuy thứ tự có đôi chút khác nhưng nhìn chung kết quả trùng khớp với thư viện **pyECLAT**
Thuật toán tự cài đặt đã hoạt động đúng với mong đợi