2025-04-19 21:49

Tags: [[Data_Mining_Lab3_Report]], [[Data Mining Lab2 Report]], [[Data Mining Lab1 Report]]

Họ và Tên: Mai Phong Đăng **MSSV**: 22280008 **Lớp**: 22KDL

Data Mining Lab4 Report

Thuật toán Apriori

- Thực hiện thuật toán Apriori để tìm các **tập mục thường xuyên (frequent itemsets)** trong một tập dữ liệu giao dịch. Mục tiêu là xác định các tập hợp mặt hàng (items) xuất hiện thường xuyên cùng nhau, nhằm phục vụ cho các bài toán phân tích như phân loại, gợi ý sản phẩm, hoặc khai phá luật kết hợp. ### Các bước thực hiện:

1. **Chuẩn bị dữ liệu:**

- Đọc dữ liệu từ một DataFrame (`df`), sau đó duyệt từng dòng (transaction).

- Với mỗi dòng, loại bỏ các giá trị `NaN` và chuyển đổi thành tập hợp các item, rồi đưa vào danh sách `transactions`.

```
import numpy as np
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
from mlxtend.preprocessing import TransactionEncoder
from collections import Counter
#Load dữ liệu
df = pd.read_csv('/content/dataW4.csv', header=None)
print(df)

transactions = []
for i in range(len(df)):
transaction = df.iloc[i].dropna().tolist()
transactions.append(set(transaction))
```

2. **Tìm tập 1-itemset xuất hiện thường xuyên (L1)**

- Đếm số lần xuất hiện từng item và lọc theo ngưỡng hỗ trợ (`min_support`).

```
item_counts = {}
for transaction in D:
    for item in transaction:
        item_counts[item] = item_counts.get(item, 0) + 1

L[1] = {frozenset([item]) for item, count in item_counts.items()
        if count / total_transactions >= min_support}
```

3. **Sinh tập ứng viên Ck từ Lk-1**

- Dựa vào thuật toán **apriori_gen**, sinh các tập ứng viên k-itemset từ các tập thường xuyên (k-1)-itemset.

```
def apriori_gen(Lk_minus_1):
    candidates = set()
    Lk_minus_1 = list(Lk_minus_1)

    for i in range(len(Lk_minus_1)):
        for j in range(i + 1, len(Lk_minus_1)):
            l1 = sorted(list(Lk_minus_1[i]))
            l2 = sorted(list(Lk_minus_1[j]))

            if l1[:-1] == l2[:-1] and l1[-1] < l2[-1]:
                candidate = frozenset(l1 + [l2[-1]])
                if not has_infrequent_subset(candidate, Lk_minus_1):
                    candidates.add(candidate)

    return candidates
```

4. **Lọc tập ứng viên theo min_supp**

- Đếm số lần xuất hiện của từng ứng viên trong dataset.

```
count_map = {candidate: 0 for candidate in Ck}
for transaction in D:
    for candidate in Ck:
        if candidate.issubset(transaction):
            count_map[candidate] += 1

Lk = {itemset for itemset, count in count_map.items()
      if count / total_transactions >= min_support}
```

5. **Kiểm tra tập con không thường xuyên**

- Dùng nguyên tắc downward closure để loại bỏ tập không hợp lệ.

```
def has_infrequent_subset(candidate, Lk_minus_1):
    k = len(candidate)
    for subset in combinations(candidate, k - 1):
        if frozenset(subset) not in Lk_minus_1:
            return True
    return False
```

6. Tổng hợp kết quả

- Gom toàn bộ các tập itemset thường xuyên từ các mức k vào một tập hợp.

```
all_frequent_itemsets = set()
for level in L.values():
    all_frequent_itemsets.update(level)
```

7. **Thống kê tần suất từng item:**

```
from collections import Counter

flat_items = [item for t in transactions for item in t]
counts = Counter(flat_items)

for item, count in counts.items():
    print(f"{item}: {count} times, support = {count/len(transactions):.2f}")
```

Kết quả đạt được

- Danh sách tất cả các tập mục thường xuyên (frequent itemsets) đã được tìm ra.
- Thống kê chi tiết số lần xuất hiện và support của từng item giúp kiểm tra tính hợp lý của thuật toán và đánh giá các mức độ phổ biến của các item.