Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ Thông tin

TÀI LIỆU LÝ THUYẾT KHAI THÁC WEB

Chủ đề 2

KHAI THÁC LUẬT KẾT HỢP & MẪU TUẦN TỰ (PHẦN 2)

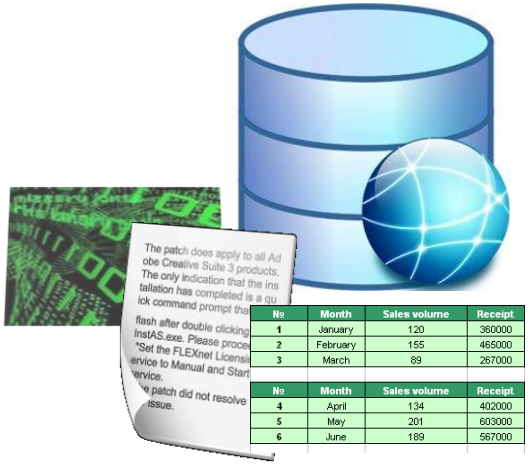
Giảng viên: ThS. Nguyễn Ngọc Thảo
Email: nnthao@fit.hcmus.edu.vn

HKII/2011-2012

NỘI DUNG

- Các vấn đề mở rộng của bài toán khai thác luật kết hợp
  - Định dạng dữ liệu cho bài toán
  - Khai thác với nhiều độ hỗ trợ tối thiểu
  - Khai thác luật kết hợp lớp

ĐỊNH DẠNG DỮ LIỆU



BIẾN ĐỔI DỮ LIỆU

- Khai thác luật kết hợp còn có thể thực hiện trên dữ liệu bảng quan hệ.
- Kiểu dữ liệu rời rạc: Chuyển từng giá trị trong bảng quan hệ thành hạng mục trong tập giao dịch dưới dạng cặp thuộc tính – giá trị.

BIẾN ĐỔI DỮ LIỆU

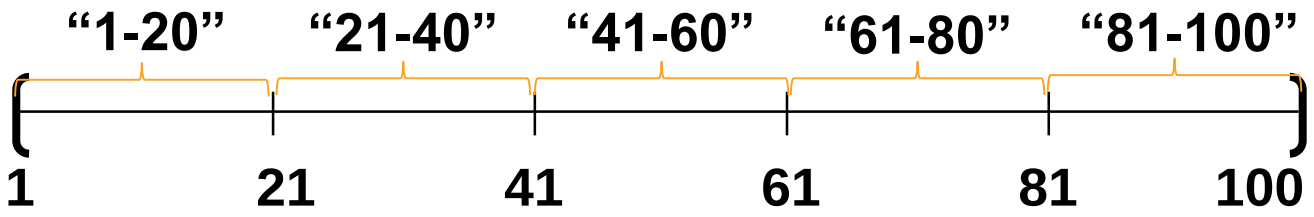
- Ví dụ biến đổi dữ liệu rời rạc
  - Từ dữ liệu bảng

Attibute1	Attribute2	Attibute3
a	a	x
b	n	y

  - Chuyển thành dữ liệu giao dịch
    - t_1: {(Attribute1.a), (Attribute2.a), (Attribute3.x)}
    - t_2: {(Attribute1.b), (Attribute2.n), (Attribute3.y)}

XỬ LÝ KIỂU DỮ LIỆU SỐ

- Kiểu dữ liệu số: rời rạc hóa miền giá trị thành đoạn và xem mỗi đoạn như một giá trị rời rạc.
  - Ví dụ:



- Có thể rời rạc hóa bằng thủ công dựa vào tri thức chuyên gia hay tự động.
  - Hiện nay có nhiều thuật toán rời rạc hóa tự động.

NHẬN XÉT

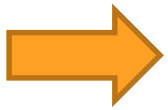
- Đối với dữ liệu bảng, cần chỉnh sửa bước gia nhập của hàm phát sinh ứng viên để không tạo ra ứng viên chứa hai hạng mục của cùng một thuộc tính.
  - Ví dụ: khi kết hợp {(Attribute1.a), (Attribute2.n)} và {(Attribute1.a), (Attribute2.b)}

NHẬN XÉT

- Ta cũng có thể chuyển từ dữ liệu giao dịch sang dữ liệu bảng.
  - Xem mỗi hạng mục trong *I* là thuộc tính.
  - Biểu diễn nhị phân: nếu giao dịch chứa hạng mục thì thuộc tính bằng 1, ngược lại bằng 0.

CSDL *T*

TID	Transaction
10	{Bread, Cheese, Juice}
20	{Milk, Bread, Yogurt}
30	{Bread, Juice, Milk}
40	{Eggs, Bread, Cheese, Juice}
50	{Cheese, Juice, Milk}



Dữ liệu nhị phân hóa

B	C	E	J	M	Y
1	1	0	1	0	0
1	0	0	0	1	1
1	0	0	1	1	0
1	1	1	1	0	0
0	1	0	1	1	0

KHAI THÁC VỚI NHIỀU ĐỘ HỖ TRỢ TỐI THIỂU

http://www.ttdesign.com/c-nov8.htm

VẤN ĐỀ KHI KHAI THÁC LKH

- Ngưỡng độ hỗ trợ tối thiểu *minsup*: thu hẹp không gian tìm kiếm và hạn chế số tập hạng mục phổ biến và luật phát sinh.

  $\Rightarrow$ Bài toán khai thác trở nên đơn giản và thực tế.

- Sử dụng ngưỡng đơn: giả sử mọi hạng mục có cùng bản chất và/hoặc có tần số xuất hiện tương tự nhau trong CSDL.

  $\Rightarrow$ Không thường đúng với ứng dụng thực tế.

BÀI TOÁN HẠNG MỤC HIẾM

- Bài toán hạng mục hiếm (rare item problem): Khi độ phổ biến của các hạng mục khác biệt nhau lớn thì
  1. Nếu minsup quá cao thì không tìm được luật liên quan đến hạng mục hiếm hay ít phổ biến.
  2. Để tìm luật liên quan hạng mục hiếm lẫn phổ biến, minsup phải đặt rất thấp.

     $\Rightarrow$ bùng nổ tổ hợp vì các tập phổ biến liên kết với nhau theo nhiều cách có thể.

BÀI TOÁN HẠNG MỤC HIẾM

- Ví dụ mua hàng trong siêu thị
  - FoodProcessor và CookingPan là mặt hàng có lãi cao nhưng không được mua thường xuyên.
  - Để tìm luật liên quan đến chúng, minsup phải rất nhỏ. Giả sử minsup = 0.005% thì tìm được

    {FoodProcessor, CookingPan} [sup = 0.006%]

  - Tuy nhiên, có 2 tập hạng mục vô nghĩa xuất hiện
    - f_1: {Bread, Cheese, Egg, Bagel, Milk, Sugar, Butter} [sup = 0.007%],
    - f_2: {Bread, Egg, Milk, CookingPan} [sup = 0.006%].

PHÂN VÙNG DỮ LIỆU

- Phân vùng dữ liệu thành nhiều khối (tập con) có kích thước nhỏ hơn, mỗi khối chỉ chứa hạng mục có tần số tương tự nhau.
- Quá trình khai thác áp dụng riêng rẽ cho từng khối với giá trị minsup khác nhau.

  $\Rightarrow$ không thỏa đáng vì sẽ không tìm được tập hạng mục (hay luật) liên quan hai khối.

SỬ DỤNG NHIỀU MINSUP

- Người dùng chỉ định độ hỗ trợ tối thiểu khác nhau cho từng hạng mục, gọi là độ hỗ trợ hạng mục tối thiểu MIS (Minimum Item Support).
- Đặc điểm của mô hình sử dụng nhiều minsup
  - Tìm các tập hạng mục chứa hạng mục hiếm nhưng không tạo ra quá nhiều tập hạng mục vô nghĩa khác.
  - Cho phép người dùng giới hạn phát sinh những tập hạng mục chỉ chứa một số hạng mục nào đó.

RÀNG BUỘC HIỆU SỐ HỖ TRỢ

- Để ngăn các hạng mục rất phổ biến và hạng mục hiếm xuất hiện trong cùng một tập hạng mục, ta sử dụng ràng buộc hiệu số hỗ trợ.
- Gọi *sup*(*i*) là độ hỗ trợ thật sự của hạng mục *i* trong dữ liệu. Với mỗi tập hạng mục *s*, ràng buộc hiệu số hỗ trợ là

$$\max_{i \in s}\{sup(i)\} - \min_{i \in s}\{sup(i)\} \leq \varphi$$

  - $0 \leq \varphi \leq 1$ là hiệu số hỗ trợ tối đa do người dùng chỉ định, giống nhau cho mọi tập hạng mục.

MÔ HÌNH MỞ RỘNG

- Độ hỗ trợ tối thiểu của luật được biểu diễn theo MIS của những hạng mục xuất hiện trong luật.
- Người dùng dễ dàng đặt nhiều điều kiện độ hỗ trợ cho các luật khác nhau.

MÔ HÌNH MỞ RỘNG

- Gọi MIS(*i*) là giá trị MIS của hạng mục *i*. Độ hỗ trợ tối thiểu của luật *R* là giá trị MIS nhỏ nhất trong số các hạng mục của luật.
- Luật *R*

$$i_1, i_2, \ldots, i_k \rightarrow i_{k+1}, \ldots, i_r$$

thỏa mãn độ hỗ trợ tối thiểu nếu độ hỗ trợ thực sự của luật trong tập dữ liệu lớn hơn hay bằng

$$\min(\text{MIS}(i_1), \text{MIS}(i_2), \ldots, \text{MIS}(i_r))$$

VÍ DỤ MÔ HÌNH MỞ RỘNG

- Xét tập các hạng mục *I* = {Bread, Shoes, Clothes}
- Giá trị MIS do người dùng chỉ định như sau

  MIS(Bread) = 2% MIS(Clothes) = 0.2%
  MIS(Shoes) = 0.1%
- Luật sau không thỏa mãn độ hỗ trợ tối thiểu

  Clothes $\rightarrow$ Bread [sup = 0.15%, conf = 70%]
- Luật sau thỏa mãn độ hỗ trợ tối thiểu

  Clothes $\rightarrow$ Shoes [sup = 0.15%, conf = 70%]
- Tại sao?

TÍNH CHẤT APRIORI

- Tính chất Apriori, hay bao đóng hướng xuống, là chìa khóa để tỉa nhánh trong thuật toán Apriori.
- Trong mô hình mở rộng, nếu dùng thuật toán Apriori để tìm tập phổ biến, tính chất này không còn đúng nữa.

VÍ DỤ TÍNH CHẤT APRIORI

- Xét 4 hạng mục 1, 2, 3, và 4 trong tập dữ liệu. Các độ hỗ trợ hạng mục tối thiểu là:

  MIS(1) = 10% MIS(2) = 20%

  MIS(3) = 5% MIS(4) = 6%

- Giả sử tìm được hạng mục {1, 2} có độ hỗ trợ 9%.
- Theo thuật toán Apriori, tập {1, 2} bị loại và do đó không có hai tập hạng mục {1, 2, 3} và {1, 2, 4}.
- Theo mô hình mở rộng, {1, 2, 3} và {1, 2, 4} có thể phổ biến. Nhưng nếu không bỏ {1, 2} thì vi phạm tính chất bao đóng hướng xuống.
- Cách giải quyết?

THUẬT TOÁN MS-APRIORI

- MS-Apriori tổng quát hóa thuật toán Apriori để tìm tập hạng mục phổ biến.
  - Khi chỉ có một MIS cho mọi hạng mục, MS-Apriori thu gọn thành Apriori.
- Dựa trên tìm kiếm theo mức.
- Phát sinh tập phổ biến bằng cách duyệt dữ liệu nhiều lần, nhưng lần duyệt thứ hai có sự khác biệt.

Ý TƯỞNG THUẬT TOÁN

- Ý tưởng: sắp xếp các hạng mục trong *I* theo thứ tự tăng dần của giá trị MIS.
- Thứ tự là cố định và dùng trong suốt quá trình thuật toán.
- Ví dụ:
  - Xét 4 hạng mục 1, 2, 3, và 4 trong tập dữ liệu với MIS(1) = 10%, MIS(2) = 20%, MIS(3) = 5%, MIS(4) = 6%.
  - Các hạng mục sắp theo thứ tự: 3, 4, 1, 2.

QUI ƯỚC THỨ TỰ HẠNG MỤC

- Gọi F_k là tập hợp gồm các tập k-hạng mục phổ biến.
- Mỗi tập hạng mục w có dạng {w[1], w[2],..., w[k]}, trong đó

$$\text{MIS}(w[1]) \leq \text{MIS}(w[2]) \leq ... \leq \text{MIS}(w[k]).$$

THUẬT TOÁN MS-APRIORI

```
Algorithm MS-Apriori(T, MS, φ)                    // MS stores all MIS values
1   M ← sort(I, MS);                              // according to MIS(i)'s stored in MS
2   L ← init-pass(M, T);
3   F1 ← {{l} | l ∈ L, l.coun
4   for (k = 2; Fk−1 ≠ ∅; k++
5     if k = 2 then
6       Ck ← level2-candidate-gen(L, φ)           // k = 2
7     else Ck ← MScandidate-gen(Fk−1, φ)
8     endif;
9     for each transaction t ∈ T do
10      for each candidate c ∈ Ck do
11        if c is contained in t then             // c is a subset of t
12          c.count++
13        if c − {c[1]} is contained in t then    // c without the first item
14          (c − {c[1]}).count++
15      endfor
16    endfor
17    Fk ← {c ∈ Ck | c.count/n ≥ MIS(c[1])}
18  endfor
19  return F ← ⋃k Fk;
```

Sắp xếp *I* dựa trên giá trị MIS của mỗi hạng mục được lưu trong MS.

THUẬT TOÁN MS-APRIORI

```
Algorithm MS-Apriori(T, MS, φ)                    // MS stores all MIS values
1   M ← sort(I, MS);                              // according to MIS(i)'s stored in MS
2   L ← init-pass(M, T);                          // make the first pass over T
3   F_1 ← {{l} | l ∈ L, l.coun
4   for (k = 2; F_{k-1} ≠ ∅; k++)
5     if k = 2 then
6       C_k ← level2-candidate-gen(L, φ)
7     else C_k ← MScandidate-gen(F_{k-1}, φ)
8     endif;
9     for each transaction t ∈ T do
10      for each candidate c ∈ C_k do
11        if c is contained in t then             // c is a subset of t
12          c.count++
13        if c – {c[1]} is contained in t then    // c without the first item
14          (c – {c[1]}).count++
15      endfor
16    endfor
17    F_k ← {c ∈ C_k | c.count/n ≥ MIS(c[1])}
18  endfor
19  return F ← ⋃_k F_k;
```

Duyệt dữ liệu lần đầu tiên sử dụng hàm init-pass(*M*, *T*).

HÀM INIT-PASS(M, T)

- Tham số: dữ liệu *T* và tập hạng mục sắp xếp *M*.
- Tạo ra tập hạt giống *L* để phát sinh tập hạng mục ứng viên có độ dài 1.
- Init-pass gồm hai bước
  1. Duyệt dữ liệu một lần để lưu lại giá trị đếm hỗ trợ của mỗi hạng mục.
  2. Lần theo thứ tự sắp xếp để tìm hạng mục *i* đầu tiên trong *M* thỏa MIS(*i*). Thêm *i* vào *L*. Với từng hạng mục *j* tiếp sau *i*, nếu $j.count / n \geq MIS(i)$, thêm j vào *L*.
     - *j.count: đếm hỗ trợ của j, n: tổng số giao dịch trong T.*

THUẬT TOÁN MS-APRIORI

```
Algorithm MS-Apriori(T, MS, φ)                    // MS stores all MIS values
1   M ← sort(I, MS);                              // according to MIS(i)'s stored in MS
2   L ← init-pass(M, T);                          // make the first pass over T
3   F1 ← {{l} | l ∈ L, l.count/n ≥ MIS(l)};       // n is the size of T
4   for (k = 2; Fk−1 ≠ ∅; k++)
5     if k = 2 then
6       Ck ← level2-candic
7     else Ck ← MScandidate-gen(Fk−1, φ)
8     endif;
9     for each transaction t ∈ T do
10      for each candidate c ∈ Ck do
11        if c is contained in t then               // c is a subset of t
12          c.count++
13        if c − {c[1]} is contained in t then      // c without the first item
14          (c − {c[1]}).count++
15      endfor
16    endfor
17    Fk ← {c ∈ Ck | c.count/n ≥ MIS(c[1])}
18  endfor
19  return F ← ⋃k Fk;
```

Tập 1-hạng mục phổ biến (F_1) được tính từ L.

VÍ DỤ LẦN DUYỆT THỨ 1

- Xét 4 hạng mục 1, 2, 3, và 4. MIS(1) = 10%, MIS(2) = 20%, MIS(3) = 5%, MIS(4) = 6%.
- Giả sử tập dữ liệu gồm 100 giao dịch và lần duyệt dữ liệu thứ nhất tính được đếm hỗ trợ như sau:

  {3}.*count* = 6 {4}.*count* = 3,

  {1}.*count* = 9 {2}.*count* = 25
- Vậy *L* = {3, 1, 2} và F = {{3}, {2}}.
- Hạng mục {4} không nằm trong *L* và {1} không nằm trong F_1. Tại sao?

LẦN DUYỆT THỨ K

- Với mỗi lần duyệt thứ *k* tiếp theo, thuật toán thực hiện 3 thao tác sau

1. Các tập phổ biến trong F_{k-1} ở lần duyệt thứ (k –1) được dùng để phát sinh ứng viên C_k bằng hàm MScandidate-gen(). Riêng k = 2, sử dụng hàm level2-candidate-gen().
2. Duyệt dữ liệu và cập nhật giá trị đếm hỗ trợ cho ứng viên trong C_k. Với mỗi ứng viên c cập nhật: đếm hỗ trợ của c và đếm hỗ trợ của c – {c[1]}.

   Nếu không phát sinh luật thì không cần tính c – {c[1]}.
3. Xác định tập hạng mục phổ biến F_k.

THUẬT TOÁN MS-APRIORI

```
Algorithm MS-Apriori(T, MS, φ)                    // MS stores all MIS values
1   M ← sort(I, MS);                              // according to MIS(i)'s stored in MS
2   L ← init-pass(M, T);                          // make the first pass over T
3   F1 ← {{l} | l ∈ L, l.count/n ≥ MIS(l)};       // n is the size of T
4   for (k = 2; Fk−1 ≠ ∅; k++) do
5       if k = 2 then
6           Ck ← level2-candidate-gen(L, φ)       // k = 2
7       else Ck ← MScandidate-gen(Fk−1, φ)
8       endif;
9       for each transaction t ∈ T
10          for each candidate c ∈
11              if c is contained in
12                  c.count++
13              if c – {c[1]} is contained in t then   // c without the first item
14                  (c – {c[1]}).count++
15          endfor
16      endfor
17      Fk ← {c ∈ Ck | c.count/n ≥ MIS(c[1])}
18  endfor
19  return F ← ⋃k Fk;
```

1. Phát sinh ứng viên C_k bằng hàm MScandidate-gen() hoặc level2-candidate-gen (k = 2).

THUẬT TOÁN MS-APRIORI

```
Algorithm MS-Apriori(T, MS, φ)                    // MS stores all MIS values
1    M ← sort(I, MS);                             // according to MIS(i)'s stored in MS
2    L ← init-pass(M, T);                         // make the first pass over T
3    F1 ← {{l} | l ∈ L
4    for (k = 2; Fk−1 ≠
5       if k = 2 then
6          Ck ← level
7       else Ck ← MS
8       endif;
9       for each transaction t ∈ T do
10         for each candidate c ∈ Ck do
11            if c is contained in t then              // c is a subset of t
12               c.count++
13            if c – {c[1]} is contained in t then     // c without the first item
14               (c – {c[1]}).count++
15         endfor
16      endfor
17      Fk ← {c ∈ Ck | c.count/n ≥ MIS(c[1])}
18   endfor
19   return F ← ⋃k Fk;
```

2. Cập nhật giá trị đếm hỗ trợ cho ứng viên trong C_k. Với mỗi ứng viên c cập nhật: đếm hỗ trợ của c và đếm hỗ trợ của c – {c[1]}.

THUẬT TOÁN MS-APRIORI

```
Algorithm MS-Apriori(T, MS, φ)                    // MS stores all MIS values
1   M ← sort(I, MS);                              // according to MIS(i)'s stored in MS
2   L ← init-pass(M, T);                          // make the first pass over T
3   F1 ← {{l} | l ∈ L, l.count/n ≥ MIS(l)};       // n is the size of T
4   for (k = 2; Fk−1 ≠ ∅; k++) do
5     if k = 2 then
6       Ck ← level2-candidate-gen(L, φ)           // k = 2
7     else Ck ← MScandidate-gen(Fk−1, φ)
8     endif;
9     for each transaction t ∈ T do
10      for each candidate c ∈ Ck do
11        if c is contained in t then             // c is a subset of t
12          c.count++
13        if c – {c[1]} is contained in t then    // c without the first item
14          (c – {c[1]}).count++
15      endfor
16    endfor
17    Fk ← {c ∈ Ck | c.count/n ≥ MIS(c[1])}
18  endfor
19  return F ← ⋃k Fk;
```

3. Xác định tập hạng mục phổ biến F_k.

LEVEL2-CANDIDATE-GEN

- Hàm level2-candidate-gen: nhận tham số L và trả về tập 2-hạng mục ứng viên.

Function level2-candidate-gen(L, φ)

```
1  C2 ← ∅;                                   // initialize the set of candidates
2  for each item l in L in the same order do
3      if l.count/n ≥ MIS(l) then
4          for each item h in L that is after l do
5              if h.count/n ≥ MIS(l) and |sup(h) – sup(l)| ≤ φ then
6                  C2 ← C2 ∪ {{l, h}};   // insert the candidate {l, h} into C2
```

- Lưu ý: sử dụng $|sup(h) - sup(l)| \leq \varphi$ vì $sup(l)$ có thể lớn hơn $sup(h)$, mặc dù $MIS(l) \leq MIS(h)$.

LEVEL2-CANDIDATE-GEN

- Ví dụ: xét các hạng mục 1, 2, 3, và 4
  - Giá trị MIS:

    MIS(1) = 10% MIS(2) = 20%

    MIS(3) = 5% MIS(4) = 6%
  - Giả sử xét trên tập dữ liệu gồm 100 giao dịch. Giá trị đếm hỗ trợ là {3}.*count* = 6, {4}.*count* = 3, {1}.*count* = 9 và {2}.*count* = 25. φ = 10%
  - L = {3, 1, 2} và F = {{3}, {2}}.
  - Hàm level2-candidate-gen tạo ra C_2 = {{3, 1}}
  - {1, 2} và {3, 2} không phải là ứng viên. Tại sao?

LEVEL2-CANDIDATE-GEN

- Ta phải dùng L thay vì F_1 do F_1 không chứa các hạng mục có thể thỏa MIS của hạng mục đứng trước nó (theo thứ tự) nhưng không thỏa MIS của chính nó.
  - Ví dụ: hạng mục 1 trong ví dụ trước.

- Vấn đề vi phạm tính chất bao đóng hướng xuống được giải quyết cho C_2.

HÀM MSCANDIDATE-GEN

- Hàm MScandidate-gen gồm 2 bước:
  - Gia nhập (join step): giống Apriori, kết hợp hai tập (k-1)-hạng mục phổ biến để tạo ra ứng viên tiềm năng c.
    - Tập hạng mục phổ biến f_1 và f_2 có các hạng mục hoàn toàn giống nhau trừ hạng mục cuối cùng.
    - c được bổ sung vào tập ứng viên C_k.
  - Tỉa nhánh (prune step)

HÀM MSCANDIDATE-GEN

- Hàm MScandidate-gen gồm 2 bước:
  - Gia nhập (join step)
  - Tỉa nhánh (prune step): với mỗi tập con *s* kích thước (*k*-1) của c, nếu *s* không nằm trong F_{k-1} thì *c* có thể được xóa khỏi C_k.

    Ngoại lệ: nếu *s* không chứa *c*[1] (chỉ có 1 tập *s* như thế), thì ngay cả khi *s* không thuộc F_{k-1}, ta không thể xóa *c.*

    Giải thích: mặc dù đã biết s không thỏa MIS(*c*[2]), ta không thể khẳng định *s* không thỏa MIS(*c*[1]), trừ khi MIS(*c*[2]) = MIS(*c*[1]).

VÍ DỤ MSCANDIDATE-GEN

- Gọi F_3 = {{1, 2, 3}, {1, 2, 5}, {1, 3, 4}, {1, 3, 5}, {1, 4, 5}, {1, 4, 6}, {2, 3, 5}}. Các hạng mục được sắp xếp thứ tự.
- Giả sử bỏ qua ràng buộc hiệu số độ hỗ trợ.
- Sau bước gia nhập:

  {1, 2, 3, 5}, {1, 3, 4, 5} và {1, 4, 5, 6}.
- Sau bước tỉa nhánh: {1, 2, 3, 5}, {1, 3, 4, 5}
  - Bỏ {1, 4, 5, 6}. Tại sao?
  - Không bỏ {1, 3, 4, 5}. Tại sao?

HÀM MSCANDIDATE-GEN

- Hàm MScandidate-gen giải quyết vấn đề vi phạm tính chất bao đóng hướng xuống cho C_k ($k > 2$).
- Giải thích: do đã sắp xếp, ta không cần mở rộng tập (k-1)-hạng mục phổ biến với hạng mục nào có giá trị MIS thấp hơn.

VÍ DỤ MS-APRIORI

TID	Transaction
t1	Beef, Bread
t2	Bread, Clothes
t3	Bread, Clothes, Milk
t4	Cheese, Boots
t5	Beef, Bread, Cheese, Shoes
t6	Beef, Bread, Cheese, Milk
t7	Bread, Milk, Clothes

CSDL *T*

MIS(Milk) = 50%
MIS(Bread) = 70%
25% cho các hạng mục khác

- F_1 = {{Beef}, {Cheese}, {Clothes}, {Bread}}
- F_2 = {{Beef, Cheese}, {Beef, Bread}, {Cheese, Bread}, {Clothes, Bread}, {Clothes, Milk}}
- F_3 = {{Beef, Cheese, Bread},{Clothes, Milk, Bread}}

CÁCH XÁC ĐỊNH MIS

- Gán một giá trị MIS cho mỗi hạng mục tùy vào độ hỗ trợ/tần số trong cơ sở dữ liệu *T*.
  - Ví dụ: MIS(*i*) = $\lambda \times sup(i)$ $(0 \leq \lambda \leq 1)$
- Gom các hạng mục thành nhóm (cluster). Các hạng mục trong một nhóm có tần số tương tự và được gán cùng giá trị MIS.
  - Mô hình mở rộng có thể tìm tập hạng mục phổ biến chứa các hạng mục thuộc nhiều cluster.

CHỌN LỰA HẠNG MỤC

- Mô hình mở rộng cho phép chọn hạng mục để phát sinh tập hạng mục phổ biến
  - Chỉ chứa những hạng mục được chỉ định
  - Hoặc không phát sinh tập hạng mục nào chỉ chứa những hạng mục khác.
- Ví dụ: từ tập dữ liệu *T* tại slide 40, đặt MIS(Beef) = 101% và MIS(Clothes) = 101%.
  - Tập phổ biến chứa ít nhất một hạng mục trong {Boots, Bread, Cheese, Milk, Shoes}
  - Không phát sinh tập chỉ chứa Beef và/hoặc Clothes

PHÁT SINH LUẬT KẾT HỢP

- Đối với MS-Apriori, chỉ lưu đếm hỗ trợ của mỗi tập hạng mục phổ biến là không đủ.
- Ví dụ:
  - Cho MIS(Bread) = 2%, MIS(Clothes) = 0.2%, MIS(Shoes) = 0.1%, *sup*({Clothes, Bread}) = 0.15%, *sup*({Shoes, Clothes, Bread}) = 0.12%
  - {Clothes, Bread} không phổ biến nhưng {Shoes, Clothes, Bread} phổ biến.
  - Không thể tính độ tin cậy cho luật Clothes, Bread $\rightarrow$ Shoes vì không có đếm hỗ trợ của {Clothes, Bread}.

BỔ ĐỀ HẠNG MỤC ĐẦU

- Bổ đề: tình trạng trên chỉ xảy ra khi hạng mục có MIS nhỏ nhất trong tập hạng mục nằm ở phần hệ quả của luật.
- Ta gọi đây là bài toán hạng mục đầu (head-item problem).

BỔ ĐỀ HẠNG MỤC ĐẦU

- Gọi *f* là tập phổ biến và $\alpha \in f$ là hạng mục có MIS thấp nhất trong *f* (α gọi là hạng mục đầu). Vậy *f* sử dụng MIS(α) làm minsup.
- Ta cần xây dựng luật $X \rightarrow Y$, trong đó $X, Y \subset f$, $X \cup Y = f$ và $X \cap Y = \varnothing$.
- Giả sử vấn đề xảy ra khi $\alpha \in X$. Vì $\alpha \in X$ và $X \subset f$, α phải có MIS nhỏ nhất trong X và X phải là tập phổ biến theo thuật toán MS-Apriori.
- Do đó, đếm hỗ trợ của X chắn chắn đã ghi nhận.

PHÁT SINH LUẬT KẾT HỢP

- Đếm hỗ trợ của $f - \{a\}$ được tính tại dòng 13-14 của thuật toán Apirori.
- Hàm phát sinh luật được thiết kế tương tự như genRules().

KHAI THÁC LUẬT KẾT HỢP LỚP

ĐẶT VẤN ĐỀ

- Luật kết hợp thông thường không hướng đến mục tiêu cụ thể.
  - Một hạng mục bất kỳ có thể nằm ở tiền đề hay hệ quả của luật.
- Trong một số ứng dụng, người dùng chỉ quan tâm đến một số hạng mục cố định nằm ở vế phải.
  - Ví dụ: cho tập hợp văn bản thuộc một số chủ đề (hạng mục mục tiêu), người dùng cần biết những từ nào tương ứng với từng chủ đề.

LUẬT KẾT HỢP LỚP

- Gọi T là dữ liệu giao dịch gồm n giao dịch. Mỗi giao dịch được gán nhãn lớp y.
- Gọi I là tập hợp các hạng mục trong T, Y là tập hợp các nhãn lớp (hay hạng mục mục tiêu) và $I \cap Y = \emptyset$.
- Luật kết hợp lớp (Class Association Rule) có dạng

$$X \rightarrow y, \text{ trong đó } X \subseteq I \text{ và } y \in Y$$

LUẬT KẾT HỢP LỚP

- Định nghĩa support và confidence giống như trong luật kết hợp thông thường.
- Luật kết hợp lớp (CAR) khác với luật kết hợp thông thường (AR) ở hai điểm:
  1. Hệ quả của CAR chỉ có một hạng mục, còn hệ quả của AR có số lượng hạng mục bất kì.
  2. Đối với CAR: hệ quả $y \in Y$, hạng mục từ I không được làm hệ quả, và nhãn lớp không được xuất hiện trong tiền đề. Trong AR, một hạng mục có thể thuộc hệ quả hay tiền đề.

BÀI TOÁN KHAI THÁC CARs

- Bài toán khai thác luật kết hợp lớp: phát sinh tập hợp đầy đủ các luật kết hợp lớp sao cho thỏa mãn ràng buộc minsup và minconf do người dùng định trước.

KHAI THÁC CARs

- Dữ liệu các tài liệu văn bản

doc 1: Student, Teach, School	: Education
doc 2: Student, School	: Education
doc 3: Teach, School, City, Game	: Education
doc 4: Baseball, Basketball	: Sport
doc 5: Basketball, Player, Spectator	: Sport
doc 6: Baseball, Coach, Game, Team	: Sport
doc 7: Basketball, Team, City, Game	: Sport

- Tập I và Y bao gồm những phần tử nào?
- Cho *minsup* = 20% và *minconf* = 60%. Hai luật kết hợp lớp có thể có là :

Student, School $\rightarrow$ Education [sup=2/7, conf=2/2]

Game $\rightarrow$ Sport [sup=2/7, conf=2/3]

KHAI THÁC CARs

- Luật kết hợp lớp có thể được khai thác trực tiếp trong một bước.
- Mục tiêu: tìm mọi hạng mục luật (ruleitem) có độ hỗ trợ lớn hơn hoặc bằng minsup.
- Một hạng mục luật có dạng (*condset*, *y*), trong đó *condset* $\subseteq$ *I* là tập các hạng mục, *y* $\in$ *Y* là nhãn lớp.

KHAI THÁC CARs

- Đếm hỗ trợ của *condset* (*condsupCount*) là số giao dịch trong *T* chứa *condset.*
- Đếm hỗ trợ của hạng mục luật (*rulesupCount*) là số giao dịch trong *T* chứa *condset* có nhãn *y*.
- Mỗi hạng mục luật biểu diễn một luật

$$condset \rightarrow y$$

  - *support = rulesupCount / n*, *n:* số giao dịch của *T*.
  - *confidence = rulesupCount / consupCount*

KHAI THÁC CARs

- Hạng mục luật thỏa minsup gọi là hạng mục luật phổ biến.
- Hạng mục luật thỏa minconf gọi là hạng mục luật tin cậy.
- Ví dụ: xét ({Student, School}, Education)
  - Support = 2/7 (28.6%), Confidence = 100%
  - Nếu minsup = 10% thì hạng mục luật phổ biến.
  - Nếu minconf = 80% thì hạng mục luật tin cậy.
  - Ta có luật: Student, School $\rightarrow$ Education [sup = 2/7, conf = 2/2]

THUẬT TOÁN CAR-APRIORI

- Thuật toán CAR-Apriori dựa trên thuật toán Apriori với các lưu ý sau:
  - Định nghĩa tập hạng mục luật ứng viên 1 phần tử: $C_1 = \{(\{i\}, y) \mid i \in I, \text{ và } y \in Y\}$
  - Hàm CARcandidate-gen() hoàn toàn tương tự như candidate-gen() và khác một điểm là: chỉ các hạng mục luật có cùng lớp được kết với nhau bằng cách kết hợp condset.

THUẬT TOÁN CAR-APRIORI

Algorithm CAR-Apriori(T)

1 $C_1 \leftarrow$ init-pass(T); // the first pass over T
2 $F_1 \leftarrow \{f \mid f \in C_1, f.rulesupCount / n \geq minsup\}$;
3 $CAR_1 \leftarrow \{f \mid f \in F_1, f.rulesupCount / f.condsupCount \geq minconf\}$;
4 **for** ($k = 2$; $F_{k-1} \neq \emptyset$; k++) **do**
5 $C_k \leftarrow$ CARcandidate-gen(F_{k-1});
6 **for** each transaction $t \in T$ **do**
7 **for** each candidate $c \in C_k$ **do**
8 **if** $c.condset$ is contained in t **then** // c is a subset of t
9 $c.condsupCount$++;
10 **if** $t.class = c.class$ **then**
11 $c.rulesupCount$++
12 **endfor**
13 **end-for**
14 $F_k \leftarrow \{c \in C_k \mid c.rulesupCount / n \geq minsup\}$;
15 $CAR_k \leftarrow \{f \mid f \in F_k, f.rulesupCount / f.\text{condsupCount} \geq minconf\}$;
16 **endfor**
17 **return** $CAR \leftarrow \bigcup_k CAR_k$;

BÀI TẬP CAR-APRIORI

TID	Transaction	Class
doc 1	Student, Teach, School	Education
doc 2	Student, School	Education
doc 3	Teach, School, City, Game	Education
doc 4	Baseball, Basketball	Sport
doc 5	Basketball, Player, Spectator	Sport
doc 6	Baseball, Coach, Game, Team	Sport
doc 7	Basketball, Team, City, Game	Sport

CSDL *T*

Minsup = 20% **Minconf = 30%**

Hãy lần lượt xác định C_k, F_k và CAR_k.

BÀI TẬP CAR-APRIORI – ĐÁP ÁN

- F_1: {({School}, Education):(3,3),
  ({Student}, Education):(2,2), ({Teach}, Education):(2,2),
  ({Baseball}, Sport):(2,2), ({Basketball}, Sport):(3,3),
  ({Game}, Sport):(3,2), ({Team}, Sport):(2,2)}
- CAR_1:
  - School → Education [sup = 3/7, conf = 3/3]
  - Student → Education [sup = 2/7, conf = 2/2]
  - Teach → Education [sup = 2/7, conf = 2/2]
  - Baseball → Sport [sup = 2/7, conf = 2/2]
  - Basketball → Sport [sup = 3/7, conf = 3/3]
  - Game → Sport [sup = 2/7, conf = 2/3]
  - Team → Sport [sup = 2/7, conf = 2/2]

BÀI TẬP CAR-APRIORI – ĐÁP ÁN

- C_2: {({School, Student}, Education),
  ({School, Teach}, Education), ({Student, Teach}, Education),
  ({Baseball, Basketball}, Sport), ({Baseball, Game}, Sport),
  ({Baseball, Team}, Sport), ({Basketball, Game}, Sport),
  ({Basketball, Team}, Sport), ({Game, Team}, Sport)}
- F_2: {({School, Student}, Education):(2, 2), ({School, Teach}, Education):(2, 2), ({Game, Team}, Sport):(2, 2)}
- CAR_2:
  - School, Student → Education [sup = 2/7, conf = 2/2]
  - School, Teach → Education [sup = 2/7, conf = 2/2]
  - Game, Team → Sport [sup = 2/7, conf = 2/2]

NHẬN XÉT MỞ RỘNG

- Nếu một hạng mục luật có confidence 100% thì khi thêm hạng mục vào condset sẽ tạo ra luật mới cũng có confidence 100% (support có thể giảm)
- Trong một số ứng dụng, những luật mới đó là dư thừa (redundant) và cần loại bỏ.

DỮ LIỆU BẢNG QUAN HỆ

- Khai thác luật kết hợp lớp với bảng quan hệ
  - Từ dữ liệu bảng

Attibute1	Attribute2	Attibute3	Class
a	a	x	positive
b	n	y	negative

  - Chuyển thành dữ liệu giao dịch
    - t_1: (Attribute1.a), (Attribute2.a), (Attribute3.x) : positive
    - t_2: (Attribute1.b), (Attribute2.n), (Attribute3.y) : negative

SỬ DỤNG NHIỀU MINSUP

1. Nhiều độ hỗ trợ lớp tối thiểu: người dùng chỉ định nhiều độ hỗ trợ tối thiểu cho các lớp khác nhau.
   - Ví dụ: chỉ định cho các luật thuộc lớp YES độ hỗ trợ tối thiểu 5%, và luật của lớp NO là 20%.
2. Nhiều độ hỗ trợ hạng mục tối thiểu: người dùng chỉ định độ hỗ trợ hạng mục tối thiểu cho từng hạng mục (nhãn lớp hoặc hạng mục thường).

SỬ DỤNG NHIỀU MINSUP

- Áp dụng thuật toán khai thác mở rộng tương tự như luật kết hợp thông thường.
- Ràng buộc hiệu số hỗ trợ cũng có thể được tích hợp.
- Ngoài ra, có thể sử dụng nhiều độ tin cậy cho các lớp khác nhau.

TỔNG KẾT

- Phương pháp chuyển dữ liệu quan hệ sang dữ liệu giao dịch và ngược lại.
- Khai thác với nhiều độ hỗ trợ tối thiểu
  - Cài đặt chương trình chạy thuật toán khai thác nhiều độ hỗ trợ MS-Apriori.
  - Thực hiện chạy tay thuật toán MS-Apriori
- Luật kết hợp lớp và cách thức khai thác luật.

TÀI LIỆU THAM KHẢO

- Tài liệu bài giảng môn học
- Chapter 2. B. Liu, *Web Data Mining-Exploring Hyperlinks, Contents, and Usage Data*, Springer Series on Data-Centric Systems and Applications, 2007.
- Chapter 5. J.Han, M.Kamber, Data Mining: Concepts & Technique, 2nd edition, Morgan Kauffman, 2006.

KẾT THÚC PHẦN 1I